OPPIMATERIAALEJA
PUHEENVUOROJA
RAPORTTEJA 111
TUTKIMUKSIA

Tero Reunanen & Sakari Koivunen (toim.)

AUTOMATISOITU MITTAUS ROBOTISOIDUSSA TUOTANTOSOLUSSA

Panoste-projektin julkaisu 4/4

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPPIMATERIAALEJA

PUHEENVUOROJA

RAPORTTEJA 111

TUTKIMUKSIA

Tero Reunanen & Sakari Koivunen (toim.)

AUTOMATISOITU MITTAUS ROBOTISOIDUSSA TUOTANTOSOLUSSA

Panoste-projektin julkaisu 4/4

Tekes

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

Panoste-projektin julkaisut:

Tero Reunanen (toim.)
Nollapistekiinnitysteknologioiden soveltaminen ja robotisoitu panostus
Panoste-projektin julkaisu 1/4
Turun ammattikorkeakoulun raportteja 108

Tero Reunanen (toim.)
Robotisoitu jäysteenpoisto
Panoste-projektin julkaisu 2/4
Turun ammattikorkeakoulun raportteja 109

Tero Reunanen & Sakari Koivunen (toim.)
Kappaleiden merkkaus robotisoidussa tuotantosolussa
Panoste-projektin julkaisu 3/4
Turun ammattikorkeakoulun raportteja 110

Tero Reunanen & Sakari Koivunen (toim.)
Automatisoitu mittaus robotisoidussa tuotantosolussa
Panoste-projektin julkaisu 4/4
Turun ammattikorkeakoulun raportteja 111

Julkaisuja koostettaessa on käytetty lähteinä seuraavien henkilöiden tekstejä:
Jani Aarnio, Dmitri Glouchenko, Tomi Grönholm, Petri Helin, Sakari Koivunen, Kalle Kuusiniemi, Matias Kylliäinen, Pessi Kääpä, Jasper Lastunen, Teemu Lehtonen, Jaakko Lento, Antti Meriö, Toni Miller, Janne Mäkelä, Marko Piira, Sami Pöllänen, Juuso Raita, Tero Reunanen, Marko Seppälä, Jouni Sirola, Kimmo Sorola, Santtu Suhonen, Pekka Törnqvist, Tommi Unkuri, Samuli Uotila, Juho Vainio, Mikko Valliluoto, Gaius Voltti ja Heikki Vuorinen.

TURUN AMMATTIKORKEAKOULUN
RAPORTTEJA 111

Turun ammattikorkeakoulu
Turku 2011

ISBN 978-952-216-197-0 (painettu)
ISSN 1457-7925 (painettu)
Painopaikka: Tampereen Yliopistopaino – Juvenes Print Oy, Tampere 2011

ISBN 978-952-216-198-7 (PDF)
ISSN 1459-7764 (elektroninen)
http://julkaisut.turkuamk.fi/isbn9789522161987.pdf

441 729
Painotuote

LUKIJALLE

Tässä julkaisussa kuvataan Panoste-projektissa tehdyt testit ja kehitetyt sovellukset sekä niiden tulokset siten, että ne olisivat mahdollisimman helposti yritysten tutustuttavissa ja käytettävissä. Julkaisu on jaettu osiin tutkimusten, testien ja ratkaisujen mukaan mahdollisimman helpon käytettävyyden saavuttamiseksi yritysten näkökulmasta, eikä niinkään yhtenäisen lukukokonaisuuden vaatimusten mukaisesti. Kirjoittamiseen on osallistunut Turun ammattikorkeakoulun opiskelijoita sekä henkilökuntaa. Osiot on koostettu pääsääntöisesti suoraan valmiista opinnäytetöistä ja raporteista. Pieniä muokkauksia teksteihin on jouduttu tekemään jotta ne on voitu koostaa yhtenäisiksi kokonaisuuksiksi tähän julkaisuun.

Tahdomme esittää kiitoksemme erityisesti Turun ammattikorkeakoulun Kone- ja tuotantotekniikan insinööriopiskelijoille jotka toimivat suuressa roolissa myös projektin dokumentoinnissa. Tämä julkaisu on pääsääntöisesti koostettu opiskelijoiden tuottamista opinnäytetöistä ja projektiraporteista. Lisäksi haluamme kiittää Pekka Törnqvistiä hänen tuestaan ja työstään julkaisun hyväksi sekä Tanja Hallenbergiä, joka on toiminut oikolukijana ja kommentoinut tekstiä.

Tero Reunanen & Sakari Koivunen

SISÄLTÖ

ESIPUHE

Vuoristoradan kyydissä. Näillä sanoilla voisi kuvailla Suomen kone- ja metalliteknologia-alan yritysten menoa Koneistettavien aihioiden uusien panostusmenetelmien käyttöönotto – Panoste -projektin aikana. Kun projektia valmisteltiin loppuvuodesta 2007 ja alkuvuodesta 2008 kävi maailman talous suurilla kierroksilla ja Suomen teollisuustuotanto oli noin 5 % kasvussa. Vuoden 2008 ensimmäisellä neljänneksellä oli metalliteollisuus suomalaisen tuotannon pääasiallisena kasvumoottorina jopa 12 % kasvuvauhdillaan. (Teollisuuden toimialakatsaus I/2008, Tilastokeskus). Kolmannella vuosineljänneksellä vuonna 2008 teollisuuden kokonaistuotannon kasvu lähes pysähtyi ja neljännellä vuosineljänneksellä alkoi jyrkkä alamäki. (Teollisuuden toimialakatsaus IV/2008, Tilastokeskus). Koko vuoden 2009 teollisuuden tuotanto supistui Suomessa tuotannon supistumisen alkaessa hidastua metalliteollisuudessa vasta vuoden 2009 viimeisellä vuosineljänneksellä. (Teollisuuden toimialakatsaus IV/2009, Tilastokeskus). Vaikka vuoden 2010 ensimmäisellä vuosineljänneksellä kaikkien teollisuuden toimialojen tuotanto oli jo kasvussa, niin metalliteollisuuden liikevaihto supistui yhä. (Teollisuuden toimialakatsaus I/2010, Tilastokeskus).

Talouden ja tuotannon vuoristorata loi projektille omat haasteensa. Vaikka Panosteen yhtenä tärkeimpänä tavoitteena oli vastata yritysten tarpeeseen selvitä juuri tällaisista tuotannon volyymin ja vaihtelevuuden muutoksista kehittämällä automatisoituja tuotantoratkaisuja kannattavuuden ja tehokkuuden parantamiseksi, niin laman vaikutuksesta johtuen projektin hyötynäkökohtien painopisteitä jouduttiin selvittämään ja ohjaamaan uudelleen projektin aikana.

Vaikka taloustilanteen vaihtelut aiheuttivat muutoksia jopa yhteistyöyritysten määrässä, niin voidaan sanoa, että Panoste onnistui silti jopa ennakko-odotuksia paremmin. Projektissa luotiin uusia sovelluksia ja ratkaisuja koneistettavien aihioiden panostukseen, jäysteenpoistoon, kappaleiden merkkaamiseen sekä tuotannon laadun valvontaan. Kaikki tutkimukset suoritettiin pitämällä mielessä konepajojen käytännön ongelmat ja kaikki sovellukset kehitettiin sellaisiksi, että pk-sektorin konepajat voivat ottaa ne käyttöönsä mahdollisimman vaivattomasti.

Panoste ei keskittynyt uusien teorioiden tai mihinkään kaukaisten visioiden luomiseen, jotka voitaisiin mahdollisesti hyödyntää vasta vuosikymmenien jälkeen. Panosteen punaisena lankana oli koko projektin läpiviennin ajan jo olemassa olevien tekniikoiden soveltaminen paremmin ja uusilla tavoilla sekä olemassa olevien sovellusten uudenlainen hyödyntäminen ja kehittäminen. Panosteen tarkoitus oli kerätä pk-yrityksiltä valitut kehityskohteet ja kehittää niihin automatisoidut ratkaisut. Tässä tehtävässä Panoste onnistui.

Turussa 24.03.2011

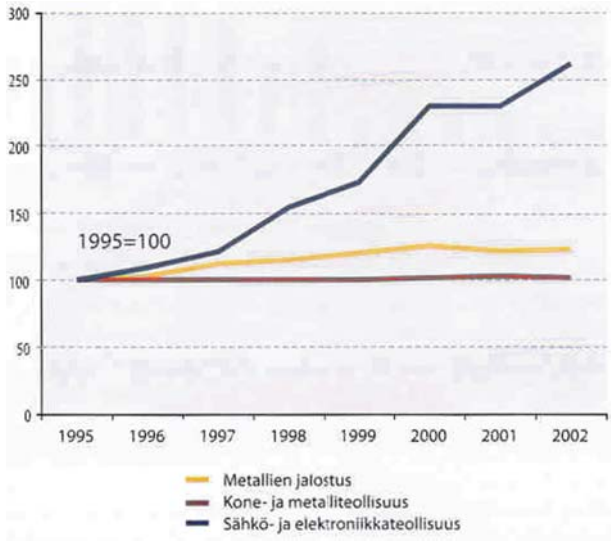
Tero Reunanen
Projektipäällikkö
Turun ammattikorkeakoulu

1 KONEISTETTAVIEN AIHIOIDEN UUSIEN PANOSTUSMENETELMIEN KÄYTTÖÖNOTTO – PANOSTE

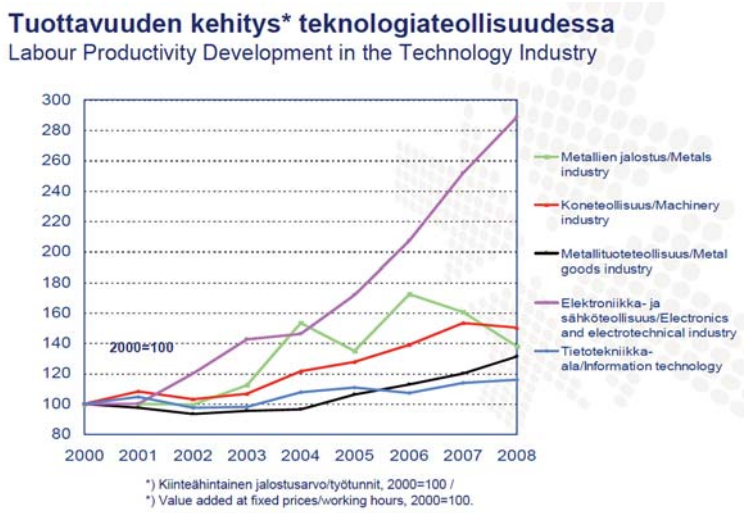
Tero Reunanen

1.1 PROJEKTIN TARVELÄHTÖISYYS JA TAVOITTEET

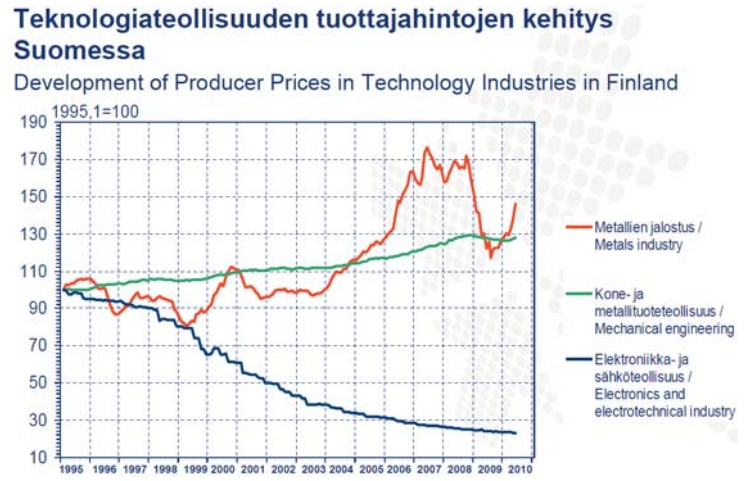
Panoste syntyi samoista lähtökohdista kuin koko SISU-ohjelma eli suomalaisen teollisuuden kilpailukyvyn heikkoudesta sekä siitä, että suomalainen kone- ja metalliteollisuus ei ollut pystynyt nostamaan tuottavuuttaan juuri lainkaan vuosina 1995–2002 (kuvio 1). Tuottavuuden kasvu vuoden 2002 jälkeen on ollut myös kohtalaisen vaatimatonta (kuvio 2), vaikka tuottavuuden kasvuun oli osaltaan vaikuttanut positiivisesti kysynnän suuruudesta johtunut hintojen nousu, jonka tuottavuuden kasvua avustava vaikutus loppui vuonna 2008 (kuvio 3).



KUVIO 1. *Koneteknologiateollisuuden tuottavuuden kehitys vuosina 1995–2002 (Teknologiateollisuus ry 2004).*



KUVIO 2. *Tuottavuuden kehitys teknologiateollisuudessa vuosina 2000–2008 (Teknologiateollisuus Ry).*

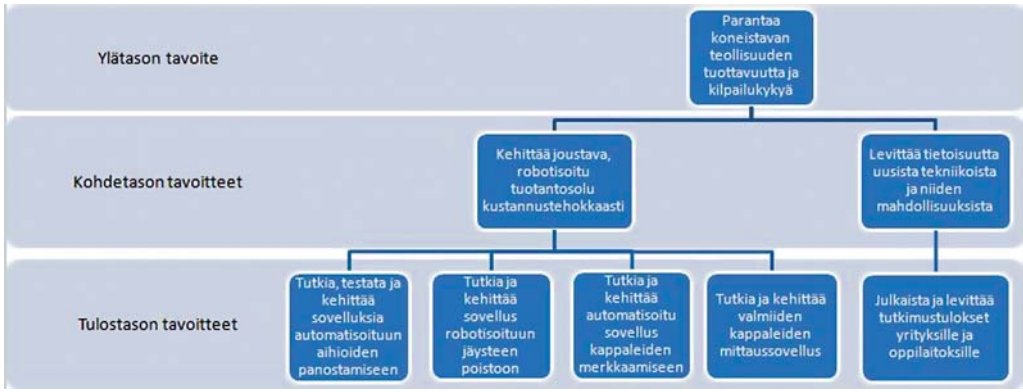


KUVIO 3. *Tuottajahintojen kehitys vuosina 1995–2010 (Teknologiateollisuus ry).*

Automatisoidun tuotannon joustavuuden parantaminen oli Panosteen toinen päätavoite ja tutkimusten kohde. Robotisointi- ja automatisointiratkaisut eivät tule ehkä koskaan olemaan niin joustavia ja adaptiivisia kuin ihminen. Toisaalta ihminen ei koskaan tule olemaan yhtä väsymätön ja tarkka kuin robotti. Ensimmäisen ohjelmoinnin jälkeen robotti pystyy toistamaan ohjelman täysin samanlaisesti pitkienkin taukojen jälkeen ilman virheitä tai muutoksia, kun taas ihmisen täytyy muistella edellistä kertaa ja tutustua uudelleen työtehtävään. Lisäksi ohjelman toistaminen onnistuu täsmälleen yhtä tarkasti toiseltakin robotilta, kun taas ihmisten välillä on suuriakin eroja muun muassa ammattitaidosta riippuen.

Suomalainen kone- ja metalliteollisuus tuottaa pääsääntöisesti pieniä sarjoja, mikä vaatii automatisoinnilta suurta joustavuutta. Tästä huolimatta automatisoidun tuotannon etuja ei voida kuitenkaan suoraan osoittaa vain sen joustavuutta tarkastelemalla. Jotta automatisoinnin kannattavuudesta ja hyödyistä saadaan todellinen kuva, tarkasteluun pitää myös ottaa mukaan muun muassa laatu-, investointikustannus- ja käyttökustannustekijät, suurin mahdollinen vuosittainen työmäärä sekä työmäärän ja työaikojen joustavuus.

Näistä ylätason tavoitteista ja rajaehdoista rajattiin projektille konkretisoidut tavoitteet. Kuvio 4 kuvaa tavoitteiden tasoa ja aktiviteetteja. Luvut 1.2.2–1.2.4 selvittävät tarkemmin yritysten tarpeiden muuntumista projektin tavoitteiksi.



KUVIO 4. *Projektin tavoitteet.*

1.1.1 Tuotannollinen näkökulma

Uusi ajattelu

Suomalainen koneistava konepajateollisuus nojautuu edelleen liikaa ”mies ja sorvi” -ajatusmalliin. Tällä tarkoitetaan ajatusta, että jokaisella koneella on oma henkilönsä tai päinvastoin, jokaisella henkilöllä on oma koneensa. Toki joissain yrityksissä on ollut myös mahdollista järjestää tuotanto siten, että 1) yksi henkilö operoi kahta tai useampaa konetta tai että 2) henkilö suorittaa koneistuksen aikana esimerkiksi jäysteen poistoa, kappaleiden merkkausta tai muita työvaiheita, joita on käsin voitu kappaleille suorittaa. Ensiksi mainittu järjestely kuvaa hyvin tuottavaa ajattelua, mutta tätäkin tuotantojärjestelyä voidaan tehostaa automatisoimalla näiden laitteiden toimintaa, jolloin henkilö pystyy operoimaan vieläkin useampia koneita yhtäaikaisesti. Jälkimmäisen vaihtoehdon ongelmana on, että siinä hukataan ammattitaitoisen työntekijän työaikaa työvaiheisiin, jotka voitaisiin automatisoida tai suorittaa vähemmän ammattitaitoisella työvoimalla.

Panosteen ajatusmaailmassa ”mies ja sorvi” on vaihtunut ”mies ja useita robotisoituja tuotantosoluja” -ajatusmalliksi. Projektin perusajatuksena ja punaisena lankana ei siis ole ollut sataprosenttiseen automaatioon tähtäävät ratkaisut, vaan

ratkaisut, joissa automatisoidaan tuotantoa niin paljon, että yksi henkilö pystyy operoimaan yhtä aikaa useita tuotantosoluja ja -laitteita.

Työnjako

Panosteessa keskityttiin myös tuotantosolujen sisäisen työjärjestyksen optimoimiseen ja suurimman tuotannollisen hyödyn saamiseen. Automaatiorajapinnan tuominen mahdollisimman lähelle koneistettavaa kappaletta mahdollistaa myös kappaleen saamisen valmiimmaksi, tai täysin valmiiksi, kerralla. Tämä vähentää yrityksen sisäistä logistiikkaa ja keskeneräisen tuotannon määrää. Kappaleen saaminen kerralla aihiosta valmiiksi tuotteeksi vapauttaa yrityksen henkilökuntaa kappaleiden siirtelystä työpisteeltä toiselle sekä pienentää keskeneräiseen tuotantoon sidottuja rahallisia resursseja.

Solujen sisäinen työnjako on suunniteltu siten, että mahdollisimman suuri osa kaikista ”avustavista töistä” voidaan suorittaa solussa, mutta koneistavan laitteen ulkopuolella robottiavusteisesti. Avustaviksi töiksi katsottiin 1) jäysteitys, 2) merkkaus ja 3) mittaus. Nämä työvaiheet arvioitiin, panostuksen lisäksi, yhdessä yritysten kanssa tärkeimmiksi automatisoinnin tutkimuskohteiksi. Projektissa kehitetyssä tuotantosolun järjestelyssä koneistava laite, joka on yleensä solun kallein laite, suorittaa vain sellaiset työvaiheet, joihin muut laitteet solussa eivät pysty. Tällöin saadaan koneistavan laitteen todellinen tuottava hyötyaika mahdollisimman suureksi. Suorittamalla avustavat työt robotilla ja Panosteessa kehitetyillä sovelluksilla saadaan myös robotin panostusten väliset odotusajat hyötykäyttöön, jolloin robotin todellinen tuottava hyötyaika kasvaa. Kun solussa suoritetaan automatisoidusti sekä panostus, jäysteitys, merkkaus että mittaus, saadaan vapautettua ammattitaitoista henkilökuntaa tuottavampaan ja ammattitaitoa paremmin hyödyntävään työhön, kuten esimerkiksi uusien kappaleiden koneistusohjelmien laatimiseen.

Tuotantovolyymin joustavuus

Tuotannon volyymin kasvattaminen ja supistaminen onnistuvat automatisoidussa tuotannossa huomattavasti helpommin ja pienemmillä ihmisiin negatiivisesti vaikuttavilla seurauksilla kuin henkilötyöhön perustuvassa tuotannossa. Kuten jo johdantokappaleessa mainittiin, kone- ja metalliteknologia-ala on ollut kovassa turbulenssissa, eikä ole nähtävissä, että tällainen suhdanteiden vaihtelu ainakaan vähenisi tulevaisuudessa.

Noususuhdanteessa tuotannon osittaisellakin automatisoinnilla voidaan tuotantovolyymia kasvattaa helposti miehittämättömällä ajolla. Miehitettyjen työvuorojen jälkeen jätetään solut valmistamaan puskurissa olevat tuotteet loppuun. Tällainen miehittämätön tuotanto ei vielä vaadi läheskään sataprosenttista automaatioastetta vaan itsenäisesti toimivan solun, johon voidaan ladata aihioita odottamaan. Miehittämättömällä työvuorolla voidaan vähentää myös ylitöiden teettämistä. Ylityöt ovat työvoimakustannuksia tarkastellen suhteellisen kustan-

nustehoton tapa lisätä tuotannon volyymia. Lisäksi jatkuvat ylityöt rasittavat yrityksen henkilökuntaa sekä fyysisesti että henkisesti. Usein myös tuotannon laatu ja tehokkuus laskevat suurien ylityökuormien aikana. Mikäli yritys toimii ympärivuorokautisesti, niin vähiten tuottavat ja raskaimmat aamuyön tunnit saattaisi olla kannattavaa hoitaa miehittämättömästi.

Laskusuhdanteessa taas voidaan jättää soluja käyttämättä. Mikäli investointi on jo kuolletettu, solun käyttämättä jättäminen ei aiheuta käytännössä lainkaan kuluja. Laskusuhdanteessa tuotannon supistamisen aiheuttamat inhimilliset ongelmat jäävät pieniksi. Automatisoidussa tuotannossa vältytään lomautuksilta ja irtisanomisilta paljon paremmin kuin ”mies ja sorvi” -tyyppisessä tuotannossa, jossa jokaisen koneen pysäyttäminen tuottaa yhden henkilön, jolle ei ole osoittaa työtehtäviä.

1.1.2 Investoinnin näkökulmat

Hankinta

Tuotannon automatisointi vaatii luonnollisesti investointinsa, mutta vaaditun investoinnin suuruutta pystytään rajaamaan muun muassa automaatioasteen järkevällä määrittelyllä sekä selvittämällä, mitä todella kannattaa automatisoida ja miten. Liian usein pyritään, ainakin myyntipuheissa ja mielikuvissa, sataprosenttiseen automaatioon. Varsinkin yksittäiskappale- ja piensarjatuotannossa sataprosenttisen automaation investointikustannukset nousevat todella suuriksi, mikäli täydellinen automatisointi on edes teknisesti mahdollista. Automatisointiastetta rajaamalla koskemaan automatisointi vain sellaisia toimia, jotka eivät vaadi liian suuria ja kalliita erikoisratkaisuja, saadaan automatisoinnin investointikustannukset pysymään siedettävinä.

Panosteen yhdeksi tavoitteeksi otettiin automatisoinnin toteuttaminen kustannustehokkaasti. Kustannustehokkuutta pyrittiin ylläpitämään sillä, että soluun kehitettävien laitteistojen ohjaaminen tapahtuu robotilla. Tällöin soluun ei tarvitse investoida ylimääräisiä ohjauskeskuksia tai muita suoraa tuottavaa työtä tekemättömiä laitteita. Ratkaisulla pyritään pitämään myös osaamiseen tarvittavat investoinnit matalampina, koska henkilökuntaa ei tarvitse kouluttaa useiden eri laitteiden käyttöön, vaan koulutus robotin ohjaamiseen riittää.

Osaaminen

Kaikkien uusien tekniikoiden käyttöönotto vaatii investointinsa myös osaamiseen, eikä koneistavan tuotannon automatisointi tee tästä poikkeusta. Mikäli yrityksessä ei jo ennestään ole robotiikan ja automaatioalan osaajia, automatisoinnin investointikustannuksiin täytyy laskea myös yrityksen miltei pakolliset investoinnit tietotaitoon. Yritys tarvitsee henkilökunnaltaan uudenlaista osaamista niin automatisoidun tuotannon käyttämisessä kuin tuotannon ja toiminnan op-

timoimisessa. Robotin ja muiden laitteiden ohjelmoiminen sekä automatisoidun tuotannon suunnittelu poikkeavat tilanteesta, jossa kaikki työvaiheet tehdään ihmisten suorittamina. Nämä uudet vaatimukset herättävät usein vastustusta automatisointia kohtaan. Tämän muutosvastarinnan voittamiseksi ja uusien tapojen omaksumiseksi henkilökunnan kouluttaminen ja heille tiedottaminen sekä henkilökunnan mukaan ottaminen automatisointihankintoja tehdessä ovat hyvin tärkeitä asioita, jotka liian usein unohdetaan.

Panoste-projektia valmisteltaessa kävi yrityksissä ilmi sekä varoittavia että hyviä esimerkkejä automatisoinnin vaatiman osaamisen saralta. Huonoissa esimerkeissä osaamiseen ei oltu investoitu tarpeeksi ja automatisointiratkaisut olivat käytössä joko huonolla hyötysuhteella tai ne oli kokonaan poistettu käytöstä. Hyvissä esimerkeissä koulutukseen oli panostettu riittävästi ja oikealla tavalla. Eräs yritys oli kouluttanut yli 55-vuotiaita henkilöitä, joilla ei ollut ennestään minkäänlaista automaatiotaustaa, robotisoitujen tuotantosolujen operaattoreiksi, jotka kykenivät itsenäisesti solujen ongelmien ratkomiseen. Panosteen eräs tavoite oli levittää tietoa automatisoinnin vaatimista taidoista sekä tarjota yrityksille esimerkkejä ja osaamisen verkottumista automatisoinnin kynnyksen madaltamiseksi.

Käyttökustannukset

Käyttökustannusten laskeminen onnistuu robotisoidussa tuotannossa suhteellisen vaivattomasti. Nykyaikaiset teollisuusrobotit ovat ehkä konepajojen toimintavarmimpia laitteita. Mikäli tuotantomäärien arviot osuvat oikeaan, laitteiston vaatimat energia- ja huoltokustannukset sekä kappaleiden vaatima tuotantoaika voidaan laskea tarkasti. Epävarmimmaksi kustannustekijäksi käyttökustannuslaskelmissa jää ihmistyövoiman osuus. Automatisoitu laitteisto tuottaa samanlaiset kappaleet täsmälleen samassa ajassa, kun taas ihmistyöhön vaikuttaa useampi tekijä.

Kuten kaikkiin laitteisiin, myös automatisoituihin järjestelmiin ja robotteihin voi tulla häiriöitä ja vikoja. Tällöin kuitenkin yhden solun vikatilanne ei seisauta kaikkea tekemistä solua hoitavalta henkilöltä. Mikäli työntekijä hoitaa esimerkiksi viittä automatisoitua tuotantosolua, yhden solun häiriö- tai rikkoutumistilanteessa hänen työpanoksestaan häviää vain 20 %, kun taas ”mies ja sorvi” -tilanteessa menetetään henkilön koko työpanos. Samalla tavalla koneiden määräaikaishuollot ja muut seisokit aiheuttavat operoivalta henkilöltä huomattavasti pienemmän työpanoksen menetyksen.

1.1.3 Teknologinen näkökulma

Panostus

Panosteen ylätason tavoitteiden saavuttaminen vaati ensisijaisesti kappaleiden panostamisen automatisoinnin tutkimista ja soveltamista. Tästä johtuen projekti nimettiinkin ”Koneistettavien aihioiden uusien panostusmenetelmien käyttöönotto”. Automatisoiduille panostusmenetelmille on yhtenäistä yksi asia: nollapiste. Kappaleisiin tarttuminen ja kappaleiden asettaminen ei ole automatisoidusti mahdollista, mikäli kappaleista ei jollain keinolla tiedetä yhtenäistä pistettä avaruuskoordinaatistossa, jonka mukaan tarttuminen ja asettaminen voidaan suorittaa. Keinoja nollapisteen aseman havaitsemiseen on useita ja ne vaihtelevat konenäöstä mekaanisiin paikoituksiin ja rajoihin. Nykyisin on markkinoilla useiden eri valmistajien useita erityyppisiä nollapistekiinnityselementtejä. Koska koko projektin perusajatus lähti panostuksen automatisoinnista, luonnollisesti *Panosteen ensimmäiseksi tutkimusalueeksi ja tavoitteeksi valittiin aihioiden robotisoidun panostuksen tutkiminen, testaaminen ja soveltaminen koneistavassa konepajatuotannossa nollapistekiinnityselementtejä hyödyntämällä*.

Jäysteitys

Jäysteitys on erittäin tärkeä koneistettujen ja sorvattujen kappaleiden viimeistelyvaihe. Jäysteen määritelmänä voidaan pitää seuraavaa: ”Jäyste on lastuavassa työstössä työkappaleen särmiin plastisen muodonmuutoksen seurauksena syntynyt ei toivottava materiaalimuodostuma.” (Gillespie 1999) Koska jäyste ei ole suunniteltua materiaalin muodostumista, jäysteen syntymistä voidaan, ja pitää, pyrkiä ensisijaisesti estämään. Jäysteen syntymistä voidaan estää muun muassa oikeanlaisella kappaleensuunnittelulla sekä oikealla työvaiheiden suunnittelulla ja järjestyksellä.

Yleensä ei silti jäysteen syntymistä voida täysin estää, vaan kappale pitää huolellisesta suunnittelusta ja oikeasta työjärjestyksestä huolimatta viimeistellä. Tähän löytyy useita eri keinoja termisestä räjäyttämisestä käsin suoritettavaan viilaukseen. Käsin suorittava jäysteenpoisto on suorittajalle suhteellisen puuduttava, yksitoikkoinen ja epämiellyttävä työvaihe. Lisäksi käsin suoritettavaan jäysteenpoistoon kuluu huomattavan paljon työaikaa ja se vaatii suorittajaltaan jatkuvaa tarkkaavaisuutta. Tämän vuoksi Panoste keskittyi jäysteen robotisoituun poistamiseen pyörivillä työkaluilla. Ratkaisulla saadaan vapautettua ammattitaitoinen työvoima tuottavampaan ja miellyttävämpään työhön ja siten voidaan hyödyntää paremmin robotin aihioiden panostusten välistä odotusaikaa. *Panosteen toiseksi tutkimusalueeksi ja tavoitteeksi tuli pyörivien jäysteitystyökalujen soveltaminen robotisoidussa tuotantosolussa sekä jäysteityslaitesovelluksen kehittäminen*.

Merkkaus

Valmistettaviin kappaleisiin halutaan usein merkintöjä, kuten tekstiä, numeroita tai kuvioita, jotka identifioivat kappaleen tai kertovat kappaleen valmistajan. Lisäksi myös kappaleen oikeanlainen käyttö saattaa vaatia merkintöjen tekemistä, kuten esim. hydrauliikkaventtiileissä, joissa halutaan kertoa venttiilin eri kanavien funktiot. Tämän lisäksi tutkimuskohteita kartoitettaessa tuli esille, että nykypäivänä kappaleiden jäljitettävyysvaatimukset ovat lisänneet kappaleiden merkkaamisen tarvetta. Useat päätoimijat tilaavat samanlaisia kappaleita eri alihankkijoilta ja vaativat tällöin alihankkijoiltaan kappaleiden täyttä seurattavuutta.

Kappaleiden merkkaaminen voi olla kohtalaisen paljon aikaa vievä prosessi. Kun Panosteen tutkimuskohteita kartoitettiin, tuli esille muun muassa seuraavanlainen tapaus: yritys valmisti hydrauliikkakomponentteja, joihin merkittiin kanavien tunnukset. Merkkaukset suoritettiin koneistuskeskuksessa pienellä terällä kaivertamalla. Kun he tutkivat asiaa myöhemmin niin kävi ilmi, että kaiverruksiin kului niin paljon aikaa, että kaikkien koneiden kaiverrusajat yhteenlaskettuna yhden koneistuskeskuksen koko vuoden työaika meni vain merkkauksien tekemiseen. Tämän tehottomuuden ymmärtää paremmin, kun käännetään ajatus niin päin, että joku ehdottaisi yrityksessä yhden koneistuskeskuksen investointia vain tämäntyyppiseen työhön. Todennäköistä on, että ehdotus ei menisi läpi ja alettaisiin etsiä muita vaihtoehtoja työvaiheen suorittamiseen. On tosin hyvin inhimillistä, että tällainen tilanne pääsee pikkuhiljaa kehittymään. Työvaihe on joskus saattanut olla järkevä suorittaa kaivertamalla koneistuskeskuksessa, mutta tuotantovolyymien kasvaessa tuotantoa ei oltu huomattu järkeistää. *Panosteen kolmanneksi tutkimusalueeksi ja tavoitteeksi tuli mahdollisimman hyvälaatuisen ja pysyvän merkkausmenetelmän soveltaminen robotisoituun tuotantosoluun kustannustehokkaasti*.

Mittaus

Panosteessa kehitettävät ratkaisut ja tuotantosolu kehitettiin sellaisiksi, että osittainen miehittämätön tuotanto on mahdollista. Osittaisella miehittämättömällä tuotannolla saadaan aikaan suurempi joustavuus ja kokonaiskäyttökustannuksia saadaan alennettua. Tämän tavoitteen saavuttamiseksi pitää kuitenkin kehittää tuotannon laatua varmistavia ratkaisuja. Mikäli laatua ei seurata mitenkään ja tuotanto pyörii miehittämättömästi, todennäköisesti jossain vaiheessa koko miehittämättömänä ajettu tuotanto on epäkuranttia ja koko tuotantoerä joudutaan siksi hylkäämään.

Panosteen valmistelussa kyseltiin yrityksien suurinta tämän hetkistä tarvetta tuotannonaikaiseen mittaamiseen, minkä tuloksena päädyttiin sorvattavien kappaleiden ulkopuolisten halkaisijoiden mittatarkkuuden varmistamiseen sekä sisäpuolisten halkaisijoiden mittaamiseen. Alkuperäisenä tarkoituksena oli kehittää yksinkertainen sovellus tuotannon pysäyttämiseksi virheellisen kappaleen ilmestyessä, mutta esiselvitysten jälkeen tavoitetta nostettiin siten, että tuotannon-

aikaisten mittaustulosten perusteella sorville syötetään parametrien korjaukset, mikäli kappaleen mitat lähestyvät toleranssirajoja. Tällöin voidaan estää ensimmäisenkin virheellisen kappaleen valmistus. *Panosteen neljänneksi tutkimusalueeksi ja tavoitteeksi muotoutui mittauslaitesovelluksen kehittäminen ulkopuolisten ja sisäpuolisten halkaisijoiden mittaamiseen*.

1.2 PROJEKTIN TIEDOT

1.2.1 SISU 2010 -ohjelma

SISU 2010 Uusi tuotantoajattelu -ohjelma aloitti toimintansa kesällä 2005 tilanteessa, jota oli edeltänyt hidas aineellisten investointien kasvu Suomen teollisuudessa sekä olematon tuottavuuden kehitys etenkin kone- ja metalliteollisuudessa. Ohjelman toteutusaikana on koettu teollisuudessa voimakkaan korkeasuhdanteen kausi, mutta myös nopeasti edennyt valmistavan teollisuuden käytännössä lamauttanut taloustaantuman vastaisku.

Ohjelman käynnistyspäätöksen taustalla vaikuttivat odotettavissa olevat merkittävät kehityspaineet valmistavassa teollisuudessa, joiden asettamiin haasteisiin ohjelmalla on haluttu vastata tukemalla kappaletavarateollisuuden ja erityisesti kone- ja metallituoteteollisuuden yrityksiä vaativissa tuotannonkehityshankkeissa ja edistämällä alan tutkimustoimintaa. Havaittiin, että halvan tuotannon maiden kilpailukyky perustuu pääasiassa edulliseen työvoimaan. Muilla kilpailukyvyn osa-alueilla suomalaisella kappaletavarateollisuudella uskottiin olevan täydet mahdollisuudet saavuttaa kansainvälinen kilpailuetu. Ohjelman missioksi määritettiinkin teknologisten edellytyksien luominen suomalaisen kappaletavarateollisuuden kansainväliselle kilpailuedulle.

Ohjelmassa on rahoitettu yhteensä yli 140 projektia, joiden yhteenlaskettu budjetti nousee 81 miljoonaan euroon. Tästä Tekesin rahoitusosuus on ollut 39 miljoonaa euroa, mistä kaksi kolmasosaa on suuntautunut yrityksiin. Yritysten hankerahoituksella on voitu pienentää kehityshankkeisiin sisältyvää riskiä yritysten näkökulmasta ja kannustaa niitä haastavampiin tuotannon kehityshankkeisiin ja suurempiin teknologiaharppauksiin. Hankerahoituksella on luotu yrityksille mahdollisuuksia erikoistumiselle ja omien toimintaprosessien ennakkoluulottomaan kehittämiseen.

Ohjelman kohderyhmänä ovat olleet teknologiateollisuuden kappaletavaroita valmistava teollisuus ja tehdastason investointihyödykkeitä tuottavat kone- ja laitevalmistajat sekä alalla toimivat tutkimuslaitokset (mukaan lukien VTT), yliopistot ja ammattikorkeakoulut. Hanketoiminta on kohdentunut erityisesti kolmelle teema-alueelle, joita ovat joustavat tuotantoratkaisut, edistykselliset tuotanto- ja valmistusteknologiat sekä itseohjautuvuus tuotannossa.

Ohjelmatoimintaan sisältyneillä seminaareilla on tuotu esille ohjelman tarjoamia palveluja ja rahoitusmahdollisuuksia sekä edistetty hankkeiden välistä vuorovaikutusta ja alan toimijoiden välistä verkottumista. Ohjelman vuosiseminaarien lisäksi on järjestetty alueseminaareja ja teemaseminaareja sekä kolme kansainvälistä seminaaria. Yhteensä seminaaritilaisuuksia on ollut 26, ja niihin on osallistunut lähes 1900 kotimaista ja yli 400 ulkomaista osallistujaa. Ulkomaisia puhujia tilaisuuksissa on ollut yli 50. (SISU 2010 Uusi tuotantoajattelu).

1.2.2 Panoste-projekti pähkinänkuoressa

Panoste toteutettiin Turun ammattikorkeakoulun ja yhteistyökumppaneiden yhteisellä ponnistuksella.

Yhteistyökumppaneita oli erityyppisiä. Osa kumppaneista oli varsinaisia projektipartnereita ja jotkut tukivat projektia mm. luovuttamalla tilojaan tai tuotteitaan projektin käyttöön. Lisäksi kumppaneiksi voitaneen laskea tavarantoimittajat, joista muutamat kuluttivat paljon aikaa ja vaivaa projektin tavoitteiden eteen. Luonnollisesti kumppaneiden panostusten määrät olivat erisuuruisia, mutta kaikkien kumppanien osallistuminen mahdollisti projektin toteuttamisen niin hyvälaatuisena kuin se toteutettiin. Panosteen projektiryhmä ja Turun ammattikorkeakoulu kiittää kaikkia kumppaneita!

Yhteistyökumppanit ja projektin rahoitus

TAULUKKO 1. *Rahoitussuunnitelma.*

Rahoitussuunnitelma		
Rahoittaja	**Summa €**	**%**
Tekes	260.818	60,00
Turun ammattikorkeakoulu	117.879	27,12
Gardner Denver Oy	5.000	1,15
Wipro Technologies Oy, Finland	5.000	1,15
ST-Koneistus Oy	5.000	1,15
Pemamek Oy	5.000	1,15
Konepaja Ceiko Oy	5.000	1,15
Carpino Oy	5.000	1,15
Salon Konepaja Oy	2.000	0,46
Mesera Works Oy	2.000	0,46
Mesera Paimio Oy	2.000	0,46
Tumo Oy	10.000	2,30
Högfors Oy	10.000	2,30
Yhteensä	**434697**	**100**

Muut partnerit:

Koneteknolgiakeskus Turku Oy (tilat ja ohjausryhmän jäsenyys)
Fastems Oy Ab (ohjausryhmän puheenjohtajuus ja asiantuntemusta)
Leinovalu Oy (asiantuntemusta)
Wärtsilä Finland Oy (ohjausryhmän jäsenyys)

Suurimman rahoitussuunnitelmassa näkymättömän ja projektilta suuria summia säästäneen panoksen projektille antoi Koneteknologiakeskus Turku Oy (KTK). Projektin tutkimuksista pääosa tehtiin KTK:n tiloissa ja laitteilla. Lisäksi projektissa valmistettujen sovellusten osat valmistettiin pääasiassa KTK:n laitteilla.

Aikataulu

Projekti aikataulutettiin alun perin aikajaksolle 1.5.2008–30.4.2010, mutta laman aiheuttamien muutosten vuoksi projektille anottiin ja saatiin lisäaikaa 31.12.2010 asti. Tämä lisäaika mahdollisti projektin suunnitellun ja tuloksellisen läpiviemisen.

Tekijät

Projekti suoritettiin sovittamalla yhteen Turun ammattikorkeakoulun insinööriopiskelijoiden opintoja, tutkimus- ja kehityshenkilöstön työtä sekä yritysten työtä. Tunneissa lasketuista työmääristä ylivoimaisesti suurimman työn suorittivat Turku ammattikorkeakoulun opiskelijat. Projektiin otti osaa yhteensä 28 opiskelijaa, jotka suorittivat projektissa opinnäytetöitään, alakohtaisia ja erikoistumisprojektiopintojaan, vapaavalintaisia opintojaan sekä työharjoittelujaan. Turun ammattikorkeakoulun henkilökunnasta aktiiviseen työskentelyyn otti osaa kahdeksan henkilöä.

1.2.3 Turun ammattikorkeakoulu

Turun ammattikorkeakoulu (AMK) on monialainen koulutusyhteisö. Siellä työskentelee 750 ammattilaista ja opiskelee 9000 opiskelijaa. Turun AMK tarjoaa työelämää ja yrittäjyyttä kehittävää koulutusta sekä organisaatioita kokonaisvaltaisesti kehittävää tutkimus- ja kehitystyötä.

Turun AMK:n seitsemästä tulosalueesta Tekniikka, ympäristö ja talous (TYT) on innovaatio-akatemia, joka vastaa alallaan Varsinais-Suomen ja osittain koko Suomen alueella korkeimman ammatillisen osaamisen kehittämisestä. Turun AMK:n ja näin ollen myös TYT-tulosalueen tärkeimmät tehtävät ovat tutkintoon johtava koulutus, aikuiskoulutus, opiskelijoiden ammatillisen kasvun tukeminen, soveltava tutkimus- ja kehitystoiminta sekä toiminta-alueensa kehittäminen.

Kukin Turun ammattikorkeakoulun tulosalueista osallistuu Tutkimus-, kehitys- ja innovaatio (TKI) -ohjelmiensa avulla toiminta-alueensa elinvoimaisuuden kehittämiseen. Turun ammattikorkeakouluun on nimetty seitsemän T&K-ohjelmaa, jotka ovat pitkäkestoisia ja tulevaisuussuuntautuneita kokonaisuuksia, joiden alla T&K-projektit toimivat. Monipuolinen yhteistyö yritysten ja yhteisöjen kanssa on tuottanut sekä lyhyitä kehittämisprojekteja että monikansallisia hankkeita. Työelämän kanssa yhteistyössä toteutettavia projekteja käynnistetään vuosittain yli 100.

1.2.4 Koneteknologiakeskus Turku Oy

Vuonna 2005 perustettu Koneteknologiakeskus Turku Oy on Varsinais-Suomen teknologiateollisuusyritysten ja Turun tekniikan alan oppilaitosten yhteinen oppimis- ja kehittämiskeskus. Turun ammattikorkeakoulun, Turun ammatti-instituutin ja Turun aikuiskoulutuskeskuksen lisäksi mukana keskuksen toiminnassa on yli 70 yritystä.

KTK tarjoaa oppilaitoksille nykyaikaiset puitteet ja laitteet teknologian opetukseen ja ammatilliseen erikoistumiseen, mahdollisuudet harjoitus- ja lopputöiden tekemiseen sekä ammattitutkintonäyttöjen suorittamiseen.

Yrityksille KTK tarjoaa uuden teknologian käyttöönottoon liittyviä koulutus- ja uudelleenkoulutuspalveluita. KTK tarjoaa yrityksille myös valmistuspalveluita, jolloin KTK:n ajanmukaista laitekantaa voidaan hyödyntää tarkoituksenmukaisesti opetusajan ulkopuolella.

Teknologiateollisuus on Turun alueen vahvin teollisuuden ala, ja sen osuus Varsinais-Suomen viennistä ja elinkeinoelämän T&K-investoinneista on huomattava. Koulutukseen hakeutuvien nuorten keskuudessa teollisuusalojen kiinnostavuus on kuitenkin vähentynyt jatkuvasti. Siksi KTK pyrkiikin esittelemään nuorille teknologia-alaa ja siihen liittyviä koulutus- ja työmahdollisuuksia sekä korjaamaan samalla osittain vanhentuneita käsityksiä alasta itsestään.

1.3 TUOTANTOSOLU

Panoste-projektin sovellukset tehtiin integroitaviksi Koneteknologiakeskus Turku Oy:ssä olevaan robotisoituun tuotantosoluun. Projektin aikana ei ollut mahdollista muuttaa solun vakiopohjapiirustusta, mutta kaikki uudet sovellukset kyettiin kuitenkin sinne sijoittamaan projektissa päätetyillä tavoilla. Lisäksi soluun tehtiin pienimuotoisia muutostöitä, kuten seinien ja latausaseman uudelleensijoittelua.

KUVA 1. *Solun ympäristö. (Kuva Pekka Törnqvist)*

KUVA 2. *Panoste-solun raakile. (Kuva Sakari Koivunen)*

1.3.1 Lastuavat laitteet

Lähteenä Pekka Törnqvistin projektiraportti.

Koneteknologiakeskuksessa on käytössä kolme työstökonetta: monitoimisorvi (Puma Series Mx 2500 ST), vaakakarainen koneistuskeskus (Doosan ACE HP 5000) ja viisiakselinen yleisjyrsinkone (Deckel Maho DMC 60 T). Koneet kuuluvat Fastems Oy:n toimittamaan FMS-järjestelmään. Panostaminen tapahtuu panostussolussa. Solussa on käytettävissä nivelvarsirobotti (Fanuc R-2000iB/165F) sekä latausasema.

1.3.2 Robotti

Lähteenä Sakari Koivusen projektiraportti.

Panoste-solussa on Fanuc R-2000iB/165F -teollisuusrobotti R-30iA -ohjauksella. Robotin kappaleenkäsittelykyky on 165 kg ja se on varustettu automaattisella työkalunvaihdolla. Sorvin panostamiseen on kahden kolmileukatarttujan yksikkö ja lisäksi nollapistekiinnikkeiden panostamiseen on oma erikoistarttujansa. Robotin 2655 mm yltämää on laajennettu asentamalla robotti 3070 mm pitkälle servotoimisille lineaariradalle. Lineaariradan ja robotin liikkeet on synkronoitu niin, että esimerkiksi suuren ja monimutkaisen kappaleen jäysteenpoistossa robottia voidaan siirtää radalla eri paikkaan jäysteenpoiston aikana.

FM-järjestelmään kytketty robottisolu voi ottaa työkappaleet suoraan hyllystöhissin varastosta. Robotin työalueella on kaksi FM-järjestelmän lavapaikkaa ja lisäksi erikoiskuljetin konepaleteille.

Konepalettia voidaan pyörittää, joten robotilla on mahdollista panostaa ja purkaa myös monitahoisia paletteja. Kappaleita valmistussoluun voidaan syöttää myös manuaalisesti kahden lavakärryn avulla.

Robotin ohjaus on kytketty Profibus-väylällä sekä ylätason FM-ohjaukseen että Daewoo -monitoimisorviin. Lisäksi robotissa on Ethernet-liitäntä esim. etädiagnostiikkaa tai, varmuuskopiointia varten. Oheislaitteiden liittämiseksi robotissa on kaksi sarjaporttia, joihin voidaan liittää esimerkiksi mittalaite tai merkkauskone.

Kun kiinnitetään kappaleita pystytasoon perinteisillä nollapiste-elementeillä, on välttämätöntä pitää kappaleesta kiinni lukitsemisen ajan. Elementin lukitseminen aiheuttaa suuria ulkoisia voimia robottiin. Tästä johtuva mekaaninen rasitus pyrittiin välttämään varustamalla robotti Softfload-optiolla. Sen avulla robotti voidaan asettaa myötäilemään ulkoisia voimia servovirtoja tarkkailemalla, joten kun elementti lukitaan, robotti seuraa perässä.

Robotissa on ohjelmistovalmiudet myös konenäön hyödyntämiseen, mutta Panosteessa konenäköä ei otettu käyttöön.

2 KONEPAJAMITTAUS

Lähteenä Antti Meriön ja Gaius Voltin projektiraportti.

Mittaus on tärkeä osa tuotantotoimintaa. Oikealla tarkkuudella suoritetun mittaamisen etuja ovat esimerkiksi tuotteiden oikea-aikainen valmistuminen, hyväksyttyjen tuotteiden vaatimustenmukaisuus, hallinnassa olevat kustannukset ja mittaustulosten hyödyntäminen tuotekehityksessä. Koska absoluuttista mittatarkkuutta ei voida saavuttaa valmistuksessa, on lanseerattu toleranssin käsite kuvaamaan mittojen sallittua vaihteluväliä. (Esala ym. 2003, 6; Ihalainen 1978, 6–8.)

2.1 PROJEKTIN MITTAUSSOVELLUKSET

Panoste-projektin mittaussovellukset on suunniteltu ratkaisemaan kaksi yleistä konepajamittauksen tarvetta: halkaisijan mittaaminen sekä ulko- että sisäpuolisista muodoista. Reunaehtona suunnittelussa pidettiin toteutuksen yksinkertaisuutta. Laitteita ohjataan robotilla, eikä tuotantosoluun tuoda erillisiä tietokoneita mittalaitteita varten. Tällä pyrittiin tuomaan kustannuksia alas sekä lisäämään käyttövarmuutta.

Ulkoisen halkaisijan mittauksessa käytetään optista mikrometriä. Mittaustapa on siis kosketukseton. Sisäpuolisen halkaisijan mittaukseen käytetään koskettavaa menetelmää. Mittaus on toteutettu sisämittalaitteella ja sarjaväylään liitettävällä mittakellolla.

2.2 JATKUVA LAADUNSEURANTA

Mittojen tarkastamisella tarkoitetaan kappaleen hyväksymistä tai hylkäämistä mittaustuloksen ja toleranssialueen perusteella. Tarkastaminen on tärkeä osa automaattista tuotantoa, mutta Panosteessa otettiin vielä yksi askel eteenpäin. Yksinkertaisesta tarkastamisesta siirryttiin prosessin ohjaamiseen mittaustulosten perusteella.

Vaikka tarkastaminen on tärkeää, se ei yksinään vielä johda optimaaliseen lopputulokseen. Esimerkkinä voidaan ajatella nopeuden säätelyä autolla ajaessa. Autoilija voi tarkastaa nopeutensa kameratolppien kohdalla. Jokainen tolpan välähdys kertoo siitä, että nopeus ei ole oikea – mitta ei ole toleranssialueella. Tämä on kuitenkin ymmärrettävästi huono ja kallis tapa. Järkevämpää on mitata no-

peutta jatkuvasti ja pyrkiä pitämään se toleranssialueen keskellä. Tämä on verrattain helppoa, koska autoissa on nopeusmittari, kaasu ja jarru. Mitataan nopeus, vertaillaan mittaustulosta tavoitearvoon ja muutetaan prosessia kaasun tai jarrun avulla. Vielä helpompaa on, jos auton varusteisiin kuuluu vakionopeudensäädin. Autolle annetaan nopeuden ohjearvo ja säädin pitää huolen siitä, että auto liikkuu annetulla nopeudella. Nopeus ei koskaan poikkea ohjearvosta suuria määriä, koska muutosta seurataan jatkuvasti ja tarvittavat korjaukset voidaan tehdä nopeasti. Panosteessa pyrittiin tuomaan vastaava konsepti koneistavaan teollisuuteen. Jokainen kappale voidaan mitata, mittatietoja seurataan tilastollisesti ja poikkeamien perusteella muutetaan koneistuksen parametreja niin, että mittatulos pysyy toleranssialueen keskellä.

3 KÄYTETTÄVISTÄ TERMEISTÄ

Lähteenä Antti Meriön projektiraportti.

3.1 MITTALAITE, MITTAUSLAITE JA MITTAUSKONE

Mittalaite ja mittauslaite

Mittauslaitteella tarkoitetaan yleisesti tietyn suureen mittaamiseen tarkoitettua yleiskäyttöistä laitetta, joka antaa numeraalisen mittaustuloksen. Mittaustulos on suoraan riippuvainen mitattavan suureen muutoksista. Tässä raportissa mittalaitteella tarkoitetaan mittauslaitetta, jolla voidaan mitata pyörähdyskappaleen ulkohalkaisija. Mittaustulos siirrettään digitaalisesti tuloksen käsittelyyn käytettävälle laitteistolle, josta sen pohjalta tarvittaessa lähetetään uusi korjausparametrin arvo työstökeskukseen.

Mittauskone

Mittauskoneella tarkoitetaan tässä yhteydessä laitteistoa, joka on valmistettu tietyn mittaustehtävän automatisointia varten. Mittauskone sisältää mittaukseen käytettävän laitteiston, esimerkiksi edellä mainitun mittalaitteen ja mitattavan kappaleen käsittelyyn ja siirtoon soveltuvan laitteiston.

Robotti

Tässä yhteydessä robotilla tarkoitetaan työstösoluun asennettua kappaleenkäsittelyrobottia.

3.2 EROTTELUKYKY JA EROTTELUKYNNYS

Erottelukyky

Mittauslaitteen tarkkuutta ja erottelukykyä eli resoluutiota ei voi suoraan vertailla. Erottelukyky voi rajoittaa tarkkuutta, mutta hyväkään erottelukyky ei takaa tarkkuutta. Erottelukyky ilmoittaa pienimmän mittaustuloksen muutoksen, jonka mittauslaite kykenee havaitsemaan. Digitaalisissa mittauslaitteissa voisi olettaa, että kysymys on vain mittaustuloksen desimaalien määrästä. Käytännössä se ei pidä paikkaansa. Tuloksen pyöristys voi tapahtua niin, että viimeinen desimaali voi olla vain tietty luku, esimerkiksi 0 tai 5. Pahimmillaan mittaustulos

voidaan ilmoittaa useammalla desimaalilla kuin todellinen erottelukynnys on. Tällöin varomaton mittaustulosten tarkastelija voi saada väärän, hyvin optimistisen käsityksen mittauslaitteen tarkkuudesta.

Erottelukynnys

Mittauslaitteen teoreettinen erottelukyky voi olla tarkempi kuin todellinen erottelukynnys. Erottelukynnys tarkoittaa pienintä mitattavan suureen muutosta, joka aiheuttaa havaittavan muutoksen mittauslaitteen tuottamaan mittaustulokseen.

3.3 MITTAUSVIRHE JA KARKEA VIRHE

Mittausvirhe

Mittauksen kokonaisvirhe eli absoluuttinen virhe sisältää sekä sisäisen että ulkoisen tarkkuuden sisältämät virheet. Vastaavasti mittaustarkkuus eli absoluuttinen tarkkuus koostuu sekä sisäisestä että ulkoisesta tarkkuudesta.

Karkea virhe

Mittaustuloksiin joskus mukaan tulevia selvästi virheellisiä tuloksia, jotka voidaan jopa silmämääräisesti todeta vääriksi, kutsutaan karkeiksi virheiksi. Käytännössä karkeat virheet voidaan jättää kokonaan huomioimatta mittaustulosta analysoitaessa, eli suodattaa pois näytteestä jo ennen mittaustuloksen analysointia. Virheen syyt voivat vaihdella, kyse voi olla esimerkiksi kirjoitusvirheestä jos mittaustulokset kirjataan ylös käsin. Tyypillisesti karkeiden virheiden syyt ovat inhimillisiä, erehdyksiä tai huolimattomuutta. Siten niiden jättäminen pois mittaustuloksia analysoitaessa ei vääristä mittaustulosta, eikä tee mittaustuloksesta epäluotettavaa.

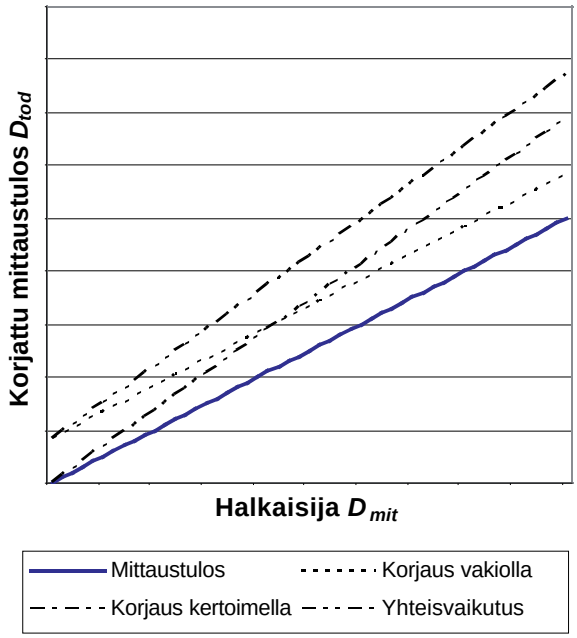
Mittaustulos ja tosiarvo

Mitattaessa saadaan tulokseksi mittaustulos, johon sisältyy aina jonkin verran virheitä. Tosiarvo puolestaan on mitattavan dimension todellinen mitta, mittaustarkkuuden rajoissa. Tosiarvo voi kuitenkin sisältää mittaustarkkuutta pienempiä virheitä, joiden vaikutus ei ole merkittävä tosiarvon luotettavuuden tai käytettävyyden kannalta.

3.4 KORJAUSARVO JA KORJAUSKERROIN

Kun käytettävän mittauslaitteiston systemaattinen virhe tunnetaan, voidaan sen korjaamiseksi määrittää matemaattinen korjausarvo tai korjauskerroin. Korjauskertoimen ja korjausarvon erona on se, onko korjauksen määrä vakio vai suhteellinen, halkaisijasta riippuva.

Vakiovirhe pysyy samana mitattavan kappaleen halkaisijasta huolimatta. Käytännössä tämä on yleensä hyvin helppoa havaita, koska virhe ilmenee jo ilman mitattavaa kappalettakin. Toisin sanoen mittalaitteen nollakohta on virheellinen. Tällöin korjausarvon ja yhteenlaskun avulla saadaan mittaustulos korjattua tarkemmaksi. Suhteellinen virhe on vaikeampi havaita, sillä nollakohta on kohdallaan. Mittaustulos poikkeaa kuitenkin esimerkiksi tietyn prosenttimäärän oikeasta arvosta. Tällöin mittaustulos voidaan korjata kertomalla se korjauskertoimella. Korjausarvon ja korjauskertoimen vaikutus esitetään kuviossa 5.



KUVIO 5. *Korjausarvon ja korjauskertoimien vaikutus.*

Koska edellä mainitut virhetyypit eivät ole toisensa poissulkevia, voi olla tarpeen käyttää sekä korjausarvoa että korjauskerrointa mahdollisimman tarkan tuloksen aikaansaamiseksi. Käytännössä mittalaitteen systemaattinen virhe on hyvin harvoin täysin lineaarinen. Tällöin pitää keskittyä tarvittavaan mittausalueeseen, ja käyttää sillä päteviä korjausarvoja ja/tai -kertoimia. Etenkin korjausarvojen kohdalla on syytä kiinnittää erityistä huomiota korjausarvon oikeaan etumerkkiin. Esimerkiksi jos havaitaan että mittaustulos on 23 yksikköä liian suuri tosiarvoon verrattuna, korjausarvo on -23 yksikköä, joka lisätään (yhteenlasku) haluttuihin korjattaviin mittaustuloksiin.

Jos virheen lähde tunnetaan, kannattaa ensisijaisesti tutkia, olisiko mahdollista eliminoida virheen lähde. Usean peräkkäisen korjauksen mahdollisuutta on syytä välttää, koska ääritilanteissa korjaukset saattavat toimia hyvinkin odottamattomalla tavalla. Siten mittauslaitteen toimintaedellytysten parantaminen ja kalibrointi ovat etusijalla mittaustulosten jälkikäsittelyyn verrattuna.

3.5 JÄLJITETTÄVYYS JA SUHTEELLINEN TARKKUUS

Jäljitettävyys tarkoittaa mittaustulosten ulkoista tarkkuutta tutkittaessa mittanormaaleista muodostuvaa vertailuketjua mittaustuloksen ja tarkempien mittanormaalien välillä. Toisin sanoen se tarkoittaa mittaustuloksen ja vertailtavien mittanormaalien välistä aukotonta ketjua, aina kansallisiin tai kansainvälisiin mittanormaaleihin asti.

Suhteellinen tarkkuus

Voidaan ajatella, että tarkkuus on suhteellinen käsite. Mittauslaitteen suhteellinen tarkkuus voidaan myös laskea vertailemalla mitattavissa olevaa maksimimittaa ja mittaustarkkuutta.

Tarkkuuden suhteellisuutta voidaan lähestyä myös toisesta näkökulmasta, suhteessa kokonaisuuteen. Yleensä esimerkiksi tuhannesosamillimetrin tarkkuus on hyvin tarkka, toisissa tilanteissa jo yhden millimetrin tarkkuus on erittäin hyvä. Tilanne riippuu siitä, mihin tehtävään tarkkuus liittyy. Myös konstruktion mittasuhteilla on merkitystä. Esimerkkinä mainittakoon rautatiekiskon mittaus: Mitattaessa rautatielinjaston pituuksia tarkkuudeksi riittää kilometrilukema, ja kiskoja käsitellään kappalemäärinä pituusmittayksikön ollessa metri. Kuitenkin ratakiskoa mitattaessa profiilin yläpinnan korkeus pitää määritellä millin kymmenyksien tarkkuudella.

Mitä suuremmista kappaleista ja mitattavista pituuksista on kysymys, sitä vaativampaa on suuren tarkkuuden saavuttaminen. Lämpölaajeneminen, epätasaisen lämpenemisen aiheuttamat lämpöjännitykset ja erilaiset kuormitukset aiheuttavat suuremman kokonaismuutoksen, kun kappaleen mitattava dimensio on suurempi.

Kosinivirhe

Kosinivirhe on merkittävä pyörähdyskappaleen ulkohalkaisijan mittaustuloksen ulkoiseen tarkkuuteen vaikuttava tekijä. Kosinivirheen aiheuttaa kulmapoikkeama anturilinjan ja kappaleen halutun mittaustason välillä.

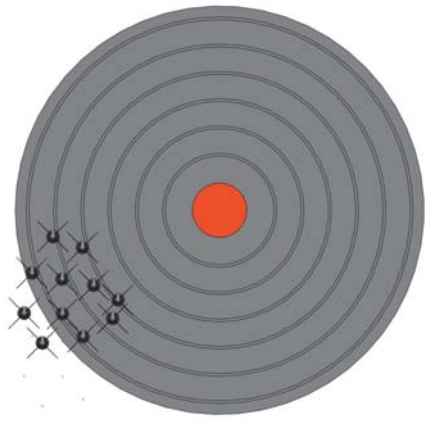
4 MITTAUSTARKKUUS

Lähteinä Sakari Koivusen opinnäytetyö ja Antti Meriön projektiraportti.

4.1 SISÄINEN TARKKUUS

Sisäinen tarkkuus eli toistuvuus (repeatability) tarkoittaa sitä, miten suuri on tarkalleen samasta kappaleesta mitattujen tulosten keskinäinen vaihtelu eli satunnainen virhe (kuva 3). Toistuvuuteen ei vaikuta mitenkään mittaustulosten ja tosiarvon välinen ero. Toistuvuutta voidaan parantaa laskemalla mittaustulos useamman mittauksen keskiarvona. Tällöin äärimmäisten tulosten vaikutus pienenee ja yksittäisten mittaustulosten jakauma lähenee normaalijakautumaa. Vaikka keskiarvoistus parantaa todennäköisyyttä välttää erittäin poikkeavia mittaustuloksia, sillä ei voida täysin korjata heikkoa sisäistä tarkkuutta. Keskiarvoistuksesta huolimatta huono sisäinen tarkkuus aiheuttaa suuren keskihajonnan mittaustuloksiin. Käytännössä se tarkoittaa suuria virherajoja mittaustuloksiin.

Sisäistä tarkkuutta voidaan analysoida tutkimalla tilastollisin menetelmin riittävän suurta mittaustulosnäytettä.



KUVA 3. *Hyvä sisäinen tarkkuus, huono ulkoinen tarkkuus.*

4.2 ULKOINEN TARKKUUS

Ulkoinen tarkkuus eli mittaustuloksen luotettavuus (accuracy) tarkoittaa mittaustulosten ja mitan tosiarvon välistä eroa eli systemaattisen virheen laajuutta. Ulkoinen tarkkuus ei ole suoraan riippuvainen sisäisestä tarkkuudesta (kuva 4). Ulkoisen tarkkuuden voi myös kuvata näytteen mittaustulosten keskiarvon ja tosiarvon välisenä erona. Jos mittaustulosten keskiarvo on hyvin lähellä tosiarvoa, niin ulkoinen tarkkuus on hyvä sisäisestä tarkkuudesta riippumatta. Keskiarvoistus ei paranna ulkoista tarkkuutta. Mittalaitteen kalibrointi on tässä paras tapa parantaa tarkkuutta.

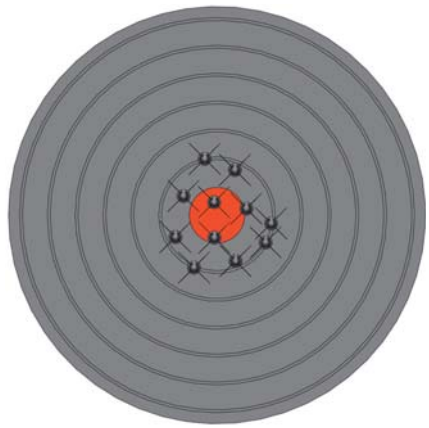
Joissakin tapauksissa ulkoista tarkkuutta voidaan parantaa mittaustuloksia matemaattisesti käsittelemällä, esimerkiksi korjauskertoimella tai korjausarvolla. Jos yhteyttä poikkeaman ja vaikuttavien tekijöiden välillä ei tarkasti tunneta, korjausta ei voida tehdä.

Ulkoiseen tarkkuuteen sisältyy monia peräkkäisiä virhemahdollisuuksia, koska käytännössä koskaan ei ole mahdollista verrata tulosta suoraan mitattavan suureen mittanormaaliin. Näiden virhemahdollisuuksien merkitys on kuitenkin hyvin pieni, kunhan käytettävissä on mitoiltaan riittävän tarkasti tunnettu referenssikappale.

Ulkoista tarkkuutta ei voida selvittää tutkimalla mittaustulosnäytettä tilastollisin menetelmin, ellei ole tiedossa mitattavan kappaleen mitan tosiarvoa.

Hyvä tarkkuus sisältää sekä hyvän sisäisen tarkkuuden että hyvän ulkoisen tarkkuuden (kuva 5).

KUVA 4. *Hyvä ulkoinen tarkkuus, huono sisäinen tarkkuus.*



KUVA 5. *Hyvä tarkkuus eli hyvä sisäinen ja ulkoinen tarkkuus.*

4.3 MITTAUKSEN EPÄVARMUUS

Koska kappaleiden valmistaminen absoluuttisella tarkkuudella ei ole mahdollista, määritellään mitoille sallittu vaihteluväli eli toleranssialue. Kappaleen mitat saavat siis poiketa nimellismitasta, mutta niiden on oltava toleranssialueen sisällä. Kappaleiden mittaustuloksia verrataan toleranssialueeseen ja mitattavat kappaleet hyväksytään tai hylätään sillä perusteella, ovatko mitat sallituissa rajoissa. Kappaleiden hyväksymisessä pitää kuitenkin ottaa myös mittalaitteen epävarmuus huomioon. Jos ollaan erittäin lähellä toleranssialueen rajaa, ei voida enää tietää, onko mittatulos hyväksymisrajojen sisäpuolella vai ei. Mittalaitteen epävarmuus pienentää valmistuksen käyttöön jäävää aluetta. Mitä epätarkempi mittalaite on käytössä, sitä tarkemmin kappaleet on valmistettava, jotta ne voidaan hyväksyä varmuudella. (Ihalainen 1978, 27–29.)

5 MITTAUSVIRHEISTÄ KÄYTÄNNÖN TILANTEISSA

Lähteinä Sakari Koivusen opinnäytetyö ja Antti Meriön projektiraportti.

5.1 VOIMIEN AIHEUTTAMAT VIRHEET

Mittauksen aikana vaikuttavat voimat aiheuttavat virheitä mittaustulokseen. Väärin suunniteltu kappaleen kiinnitys mittauksen aikana saattaa aiheuttaa muodonmuutoksia itse kappaleeseen. Koska kappaleen oma paino saattaa poikkeuttaa kappaleen muotoa, on tärkeää suunnitella kiinnitys niin, että se tukee kappaletta riittävästi. Kiinnitysvoimat eivät kuitenkaan saa nousta liian suuriksi, koska liian suuret kiinnitysvoimat johtavat muodonmuutoksiin ja mittausvirheisiin. Liian hento kiinnitys puolestaan saattaa joustaa mittausvoimien tai kappaleen painon vaikutuksesta. Tyypillisesti mittausvoimat ovat kuitenkin hyvin pieniä ja painovoimalla on mittausvoimia suurempi vaikutus. (Andersson & Tikka 1997, 133–134.)

Myös mittalaite muuttaa muotoaan mittausvoimien ja maan vetovoiman vaikutuksesta. Esimerkiksi mittakellon varsi joustaa hieman omasta painostaan. Mittalaitteeseen vaikuttavien voimien aiheuttaman virheen minimoimiseksi mittalaitteen pitää olla rakenteeltaan tukeva. Virhettä aiheuttaa myös mittakärjen painuminen mitattavaan kappaleeseen. Sillä ei tyypillisesti ole merkitystä, koska painumisen aiheuttama virhe on hyvin pieni. (Ihalainen 1978, 21; Andersson & Tikka 1997, 135.)

5.2 INHIMILLISET VIRHEET

Kun ihminen suorittaa mittauksen, pitää mittalaitteen epätarkkuuden lisäksi ottaa huomioon myös mittaajan tarkkuus ja huolellisuus. Analogisen mitta-asteikon tarkkuus on riippuvainen silmän tarkkuudesta. Tämä epätarkkuus voidaan eliminoida digitaalisella osoitinlaitteella, jolloin tulkintaepävarmuutta ei enää ole.

Parallaksipoikkeamaksi kutsutaan sitä virhettä, kun mittalaitetta vinoon katsottaessa osoitin näkyy väärässä kohdassa mitta-asteikkoon nähden (kuva 6). Esimerkiksi mittakellon lukematarkkuus kärsii, jos mittaaja ei lue tulosta kohtisuoraan mitta-asteikkoon nähden. Vino katselukulma saa osoittimen näkymään väärässä kohdassa mitta-asteikolla. (Ihalainen 1978, 26.)

Robotisoitu mittaustapahtumakaan ei täysin sulje pois inhimillisten virheiden mahdollisuutta. Siitä huolimatta odottamattomien inhimillisten virheiden vaikutus vähenee. Voidaan olettaa, että oikein ohjelmoitu robotti suorittaa mittausta luotettavasti ilman jatkuvaa valvontaa. Osa inhimillisten virheiden välttämistä on myös se, että mittauslaitteelle lähetetään toimiviksi todetut asetukset sähköisesti aina mittausohjelmiston käynnistyksen jälkeen.

Inhimillisen virheen lähde piilee myös siinä, että mittalaitteen tulokseen luotetaan ilman määräaikaisia tarkistuksia. Käytännössä on edelleen syytä tarkastusmitata työstettyjä kappaleita pistokoeluotoisesti ennalta määrätyn suunnitelman mukaisesti. Osa suunnitelmaa on myös hylättyjen kappaleiden tarkistusmittaukset. Mikäli hylättyjen kappaleiden mitat eivät noudata odotettavissa olevaa jakaumaa, on aiheellista tarkastusmitata myös hyväksyttyjä kappaleita riittävässä laajuudessa.

KUVA 6. *Parallaksivirhe.*

5.3 EPÄPUHTAUDET

Leikkuuneste ja työstölastut on poistettava koneistettujen kappaleiden pinnasta ennen mittausta. Mittaussäteen kohdalla kappaleen pinnassa olevat epäpuhtaudet vaikuttavat merkittävästi mittaustuloksen oikeellisuuteen.

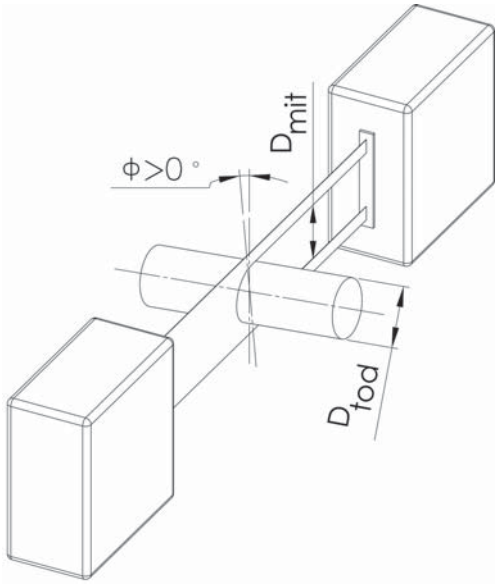
Tarvittaessa on käytettävä esimerkiksi paineilmaa kappaleen puhdistamiseen. Kappaleen muodosta ja työstöjärjestyksestä riippuen jopa pesu ja huolellinen kuivaus voi olla tarpeen riittävän mittaustarkkuuden takaamiseksi. Yleensä kuitenkin kappaleen pyörimisliike työstökoneessa linkoaa epäpuhtaudet tehokkaasti pois kappaleen pinnasta.

5.4 KOSINIVIRHE

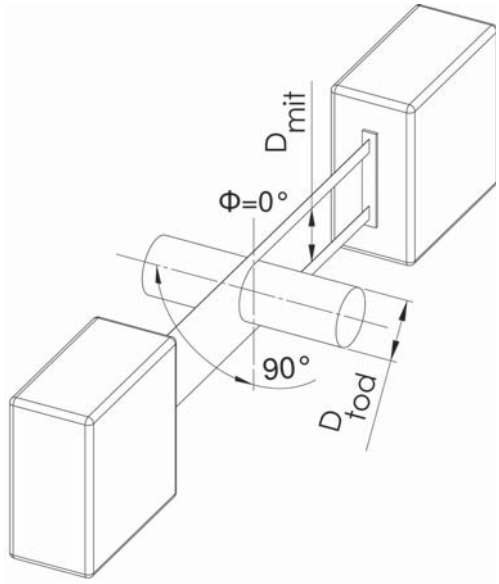
Jos mitattava kappale ja mittalaite ovat eri kulmassa toisiinsa nähden, mittatulokseen syntyy virhe. Tätä virhettä kutsutaan kosinivirheeksi. Sen syntymistä voidaan estää mittalaitteen rakenteellisilla ratkaisuilla. Esimerkiksi tasomaisilla kärjillä varustettu kaarimikrometri hakeutuu automaattisesti kohtisuoraan mitattavaan pintaan nähden. (Andersson & Tikka 1997, 136–138.)

Mitattavan kappaleen ja mittausanturien keskinäisen sijoittelun kannalta merkittävin asia on anturilinjan ja kappaleen halutun mittaustason samansuuntaisuus. Mittaustason ja anturilinjan välisen kulmavirheen, eli kun $\varphi \neq 0°$, seuraus on kosinivirhe: $D_{tod} = \cos\varphi \cdot D_{mit}$

Optimitilanteessa, kun $\varphi \neq 0°$, niin $D_{mit} = D_{tod}$. Kuva 7 havainnollistaa kosinivirhettä.



KUVA 7. *Kosinivirhe*



KUVA 8. *Optimitilanne* $D_{mit} = D_{tod}$.

On syytä huomata kosinivirheen ja halkaisijan riippuvaisuus: Virhekulman pysyessä vakiona ja kappaleen halkaisijan kasvaessa kosinivirheen vaikutus kasvaa samassa suhteessa. Esimerkiksi virhekulmalla φ =2° ja halkaisijalla D=10 mm kosinivirheen vaikutus mittaustulokseen on 6 µm, mutta halkaisijalla D=100 mm 61 µm. Vastaavat pitkittäissuuntaiset eromitat kehällä halkaisijan matkalla olisivat vastaavasti 0,349 mm ja 3,492 mm. Toisaalta, jos kulman mittaustapa ja -tarkkuus pysyvät samana, 0,349 mm pitkittäinen eromitta halkaisijalla D=100 mm aiheuttaisi 0,20 asteen kulmavirheen ja 1 µm kosinivirheen.

Kosinivirheen vaikutusta ei voi poistaa koko mittausalueelta mittalaitteen yhden pisteen mukaisella kalibroinnilla tai vakiolla korjausarvolla. Tämä koskee erityisesti nollapisteen siirrolla tapahtuvaa korjausta kalibroinnissa, eli niin sanottua offset-korjausta. Yksittäisessä pisteessä kosinivirheen vaikutusta on toki mahdollista vähentää. Kulmapoikkeamat korjauksessa vallinneesta hetkestä aiheuttavat kuitenkin edelleen kosinivirheen, mutta vaikutus on pienempi. Tämä johtuu kosini-funktion aaltomuodosta.

Mittalaite-vaihtoehdossa kehitettiin olake-kappale mittauskoneen kulmavirheen tarkastamiseen. Kappaleen olakkeen kohtaa mitattaessa mittalaitteen antamista tuloksista olisi yksinkertaista tarkasti havaita kulmavirheet.

Kosinivirheen vaikutus voidaan välttää myös iteroimalla. Valittu robotti-vaihtoehto toteuttaa iteraation periaatteen. Mitattava kappale lähestyy robotin vaaputusliikkeellä kulmaa $\varphi=0°$, jolloin mittaustulokset pienenevät. Vaaputusliikkeen ohitettua optimikulman mittaustulokset alkavat jälleen kasvaa. Mittalaitteen automaattisella minimiarvon pidolla pienin mitattu halkaisija (hetkellä $\varphi \neq 0°$) on helppoa taltioida. Vaaputusta käytettäessä on huomattava, että mittalaitteen keskiarvoistusta pitää pienentää verrattuna staattisen kappaleen mittaukseen. Suurella keskiarvoistuksella ilmoitettavaan mittaustulokseen sisältyy jo niin monia mittauksia, että mittalaitteen automaattisen minimiarvon pidon vaikutus alkaa kärsiä. Tällöin mittaustulokset alkavat kasvaa, joten suurempi keskiarvoistus aiheuttaisi systemaattisen virheen.

5.5 LÄMPÖTILA JA MUUT YMPÄRISTÖTEKIJÄT

Lämpötilan vaikutus kappaleen mittoihin on tyypillinen virhelähde tarkoissa mittauksissa. Teknisissä mittauksissa peruslämpötila on aina +20 °C ja esimerkiksi piirustusten mitat tarkoittavat mittaa +20 °C lämpötilassa, vaikka sitä ei olisikaan mainittu erikseen. (Andersson & Tikka 1997, 130.) Eri materiaalit käyttäytyvät eri tavalla lämpötilan muuttuessa. Kullekin materiaalille on määritelty lämpöpitenemiskerroin, joka kertoo, miten paljon lämpötilan muutos vaikuttaa pituuteen.

Vaikka lämpölaajeneminen ei sinänsä aiheuttaisi mitoiltaan hyväksytyn kappaleen mittojen muuttumista toleranssien ulkopuolelle, vaikutus teränkorjausparametreihin voi olla virheellinen. Lämpölaajenemisen maksimivaikutusta on syytä arvioida kappaleen halkaisijan ja lämpötilaeron suhteessa. Mikäli kokonaisvaikutus on vähäinen mittaustarkkuuteen verrattuna, se voidaan jättää kokonaan huomioimatta.

Lämpölaajenemisen kompensoinnissa on huomioitava myös toiminta tuotantokatkoksen jälkeen. Tällöin on mahdollista että mitattava kappale on ehtinyt jäähtyä katkoksen aikana lähelle ympäristön lämpötilaa. Jos työkaluparametria korjataan saadun mittaustuloksen pohjalta samoin kuin tuotantolämpötilassa olevan kappaleen kohdalla tehdään, lopputuloksena on virheellinen korjaus. Pahimmillaan virheellinen korjaus voi johtaa seuraavien työstettävien kappaleiden hylkäämiseen

Lämpötilan aiheuttama virhe on luonteeltaan systemaattinen, jos tunnetaan sekä kappaleen että mittalaitteen lämpötila ja lämpöpitenemiskerroin (Andersson & Tikka 1997, 131). Kappaleen lämpötilan mittaaminen ei aina ole yksinkertaista.

Esimerkiksi sorvatun kappaleen pintalämpötila saattaa olla koneistuksesta johtuen selvästi ydinlämpötilaa korkeampi. Koneistamon ilman lämpötila on helppo mitata, mutta sillä ei välttämättä ole mitään tekemistä kappaleen lämpötilan kanssa. Kappaleen pintalämpötila voidaan myös mitata, mutta sekään ei aina kerro koko totuutta. Lämpötilan voidaan antaa tasaantua ennen mittausta, mutta jos mittaus halutaan suorittaa välittömästi valmistuksen jälkeen, ei kappaleen lämpötila ehdi välttämättä tasaantua. Lämpölaajenemiseen vaikuttavat myös kappaleen lämpötila ennen koneistusta, koneistuksen kesto sekä odotusaika koneistuksen ja mittauksen välillä. Teräksen lämpölaajenemiskertoimen vaikutus on noin 1 µm/100 mm 1 K lämpötilanmuutoksessa. Koneistuksen aiheuttama lämpötilan muutos ei kuitenkaan ole tasainen, vaan riippuu koneistuksen aikana poistettavasta materiaalimäärästä ja kappaleen massiivisuudesta.

Muiden ympäristötekijöiden kuin lämmön vaikutus on hyvin pieni. Ilmanpaineen ja kosteuden normaalit vaihtelut sisältyvät mittauslaitteen virherajoihin valmistajan ilmoittamiin käyttöolosuhteiden rajoihin asti.

5.6 OPTISET VIRHEET

Suurin osa optisista virheistä sisältyy mittalaitteen virherajoihin. Hiukkasmaiset ilman epäpuhtaudet voivat kuitenkin aiheuttaa merkittäviä epätarkkuuksia mittaustulokseen. Siksi mittauslaite ja mitattava kappale on suojattava laskeutuvilta ilman epäpuhtauksilta. Konepajaolosuhteissa laitteiston suojaus on vähimmäisvaatimus. Jos epäpuhtauksia on enemmän, on tarpeen koteloida mittauslaite ja varustaa se puhtaan ilman syöttölaitteistolla. Pitää kuitenkin huomioida ilmavirtojen heikentävä vaikutus mittaustarkkuuteen, etenkin jos ilmavirtojen lämpötilaerot ovat suuria.

6 LAADUNVALVONTA

Lähteenä Antti Meriön projektityö.

Mittaustuloksia voidaan käyttää robotin ohjaukseen niin, että toleranssirajat ylittävät kappaleet erotetaan normaalista kappalevirrasta. Kaikki hylätyt kappaleet eivät välttämättä tule lopullisesti hylätyiksi. Esimerkiksi puhdistuksen ja tarkastusmittauksen sekä automaattisen tai manuaalisen uudelleenmittauksen jälkeen osa voidaan mahdollisesti hyväksyä.

6.1 TYÖSTÖPARAMETRIEN KORJAUKSET

Yksittäisen kappaleen mittaustuloksen pohjalta ei voi suoraan asettaa työstökeskuksen korjausparametreja. Terän rikkoutuminen, epäpuhtaudet ja muut, vaikeasti ennustettavat epävarmuustekijät voisivat silloin aiheuttaa liian nopean korjauksen muutoksen. Seurauksena tästä olisi hylättyjen kappaleiden määrän lisääntyminen.

On syytä laskea korjaus esimerkiksi liikkuvana keskiarvona, ja vähintäänkin rajoittaa pois epänormaalin suuret kertakorjaukset. Epänormaalina voidaan pitää kahden peräkkäisen mittaustuloksen muutosta, joka edellyttäisi esimerkiksi teräpalan särmän käyttökelpoista syvyyttä suurempaa kokonaismuutosta teräpalan geometriassa. Käytännössä raja on syytä asettaa selvästi pienemmäksi. Korjaukselle on myös syytä asettaa kokonaiskorjauksen maksimiraja, jonka ylittäminen ei normaaleissa olosuhteissa ole mahdollista.

Asetettujen rajojen ylittävistä lyhyt- tai pitkäaikaisista muutoksista on helppo välittää hälytysviesti eteenpäin häiriötilanteesta. Nopea muutos kappaleen mitassa, tarvittavan korjauksen epänormaali muutossuunta tai kokonaiskorjausrajan ylittyminen ovat muutamia selkeitä hälytysrajoja.

6.2 ERIKOISTILANTEET

Mittaustulosten analysoinnissa on huomioitava muutokset koneistuksen kulussa. Esimerkiksi teräpalan (tai työkalun-) vaihdon jälkeen työstökoneelle lähetetty korjaus on palautettava lähtötilanteeseen.

Mittaus voidaan haluttaessa suorittaa myös useammassa vaiheessa. Jos kappaleen lämpölaajeneminen ei ole tasaista, voidaan asettaa uusia hyväksyntäehtoja rajatapausten käsittelyyn. Jos juuri koneistettu kappale ei ole suoraan hyväksyttävissä, se voidaan asettaa sivuun ja mitata uudelleen asetetun tasaantumisajan kuluttua.

Kun kappaleen käytös lämpölaajenemisen suhteen tunnetaan, voidaan asettaa eri rajamitat kylmälle ja koneistuslämpöiselle kappaleelle.

7 TILASTOLLISET TUNNUSLUVUT

Lähteenä Sakari Koivusen opinnäytetyö.

Ihmisten on vaikea käsitellä suuria tietomääriä. Usein ongelmana onkin tietojen tulkitseminen ja ymmärtäminen, ei tiedon puute. Esimerkiksi tieto yrityksen työtekijöiden iästä on henkilöstöhallinnolla helposti saatavilla, mutta pitkä lista syntymäaikoja on hankalasti hahmotettavissa. Tiedon tulkitsemiseen tarvitaan helpottavia työkaluja.

Suuriakin tietomääriä voidaan yksinkertaistaa erilaisilla menetelmillä. Vanhan sanonnan mukaan kuva kertoo enemmän kuin tuhat sanaa – siksi graafinen esitys on usein tehokas tapa kuvata tietoa.

Tietojoukkoja voidaan havainnollistaa myös tilastollisilla tunnusluvuilla, jotka kuvaavat koko aineistoa mahdollisimman hyvin. Näin suuri tietomäärä saadaan supistettua muutamaan helposti ymmärrettävään lukuun, jotka kuitenkin antavat melko hyvän käsityksen koko tietojoukosta.

7.1 KESKILUVUT

Keskiluvut kuvaavat arvojen keskimääräistä sijaintia mitta-asteikolla. Koulunsa päättäviltä kysytään usein kurssien arvosanoja, mutta harva kysyy kaikkia arvosanoja. Halutaan kuulla vain yksi luku, joka kuvaa koulumenestystä.

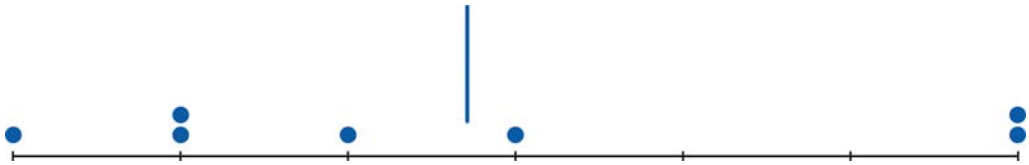
Vastaavia tilanteita on runsaasti. Usein on hyödyllistä saada yksi luku, joka antaa käsityksen suuremman tietojoukon arvojen keskimääräisestä sijainnista. Tällaisia tunnuslukuja kutsutaan keskiluvuiksi.

7.1.1 Aritmeettinen keskiarvo

Aritmeettinen keskiarvo (*arithmetic mean, mean*) on yleisin käytössä olevista keskiluvuista. Puhekielessä siihen viitataan usein sanalla keskiarvo. Aritmeettinen keskiarvo on joukon arvojen summa jaettuna joukon arvojen lukumäärällä. Se on helppo laskea ja useissa tilanteissa kuvaa joukkoa varsin hyvin. Aritmeettista keskiarvoa merkitään tyypillisesti vaakaviivalla muuttujan päällä, esimerkiksi $\bar{x}$. (Henttonen ym. 2004, 308.)

Kuvassa 9 olevan lukusarjan 0, 1, 1, 2, 3, 6, 6 aritmeettinen keskiarvo on 2,71.

$$x = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{0+1+1+2+3+6+6}{7} = 2,71$$



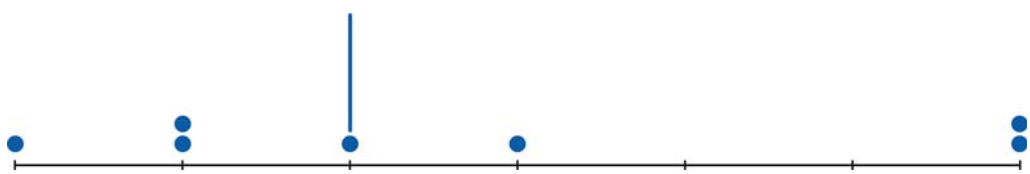
KUVA 9. *Aritmeettinen keskiarvo.*

Poikkeuksellisen suuret tai pienet arvot vaikuttavat merkittävästi keskiarvoon. On tapauskohtaista, onko tämä hyvä vai huono asia. Jos suurten arvojen vaikutuksen katsotaan vääristävän tulosta, voidaan käyttää jotain muuta keskilukua, esimerkiksi mediaania.

7.1.2 Mediaani

Mediaani (*median*) jakaa joukon keskeltä kahteen osaan. Alempaan puolikkaaseen jäävät arvot, jotka ovat mediaania pienempiä ja ylempään puolikkaaseen jää yhtä monta arvoa, jotka ovat mediaania suurempia. Kuvaan 10 merkitty mediaani on siis suuruusjärjestykseen laitetun joukon keskimmäinen arvo, kun arvoja on pariton määrä. Jos arvoja on parillinen määrä, ilmoitetaan mediaaniksi joko kaksi keskimmäistä arvoa tai niiden aritmeettinen keskiarvo. (Henttonen ym. 2004, 310.)

Joissakin tilanteissa mediaani kuvaa keskimääräistä arvoa paremmin kuin aritmeettinen keskiarvo. Esimerkiksi asuntojen keskihintaa kuvattaessa saattaa mediaani olla toimiva tunnusluku. Pienellä rannikkopaikkakunnalla voi olla myynnissä muutamia ränsistyviä rintamamiestaloja, mutta sattumalta myös miljoonien eurojen edustushuvila, ainoa laatuaan koko pitäjässä. Aritmeettinen keskiarvo johtaa erikoiseen tilanteeseen, missä loistohuvilaa lukuun ottamatta kaikki myytävät asunnot ovat aritmeettista keskiarvoa halvempia. Tässä tilanteessa mediaani kuvaa tyypillistä asunnon hintaa paremmin kuin aritmeettinen keskiarvo.

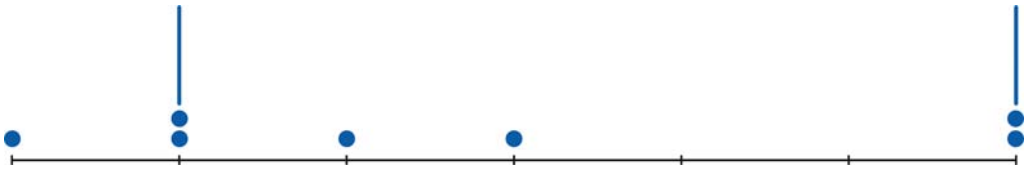


KUVA 10. *Mediaani kuvaa joukon keskimmäistä arvoa.*

7.1.3 Moodi

Moodi (*mode*) on joukossa useimmiten esiintyvä arvo eli tyyppiarvo. Moodeja voi tietojoukosta riippuen olla useita. Esimerkiksi kuvassa 13 esitetyllä lukusarjalla 0, 1, 1, 2, 3, 6, 6 on kaksi moodia: 1 ja 6. (Henttonen ym. 2004, 310.)

Moodin on hyödyllisimmillään aineistoissa, joista ei voida ottaa muita keskilukuja. Kun lemmikkejä kartoittavassa kyselyssä on vastausvaihtoehtoina ”kissa”, ”koira” ja ”muu tai ei lemmikkiä”, ei tuloksista voida laskea aritmeettista keskiarvoa. Voidaan kuitenkin määritellä yleisin vastaus eli moodi.



KUVA 11. *Joukossa voi olla monta moodia eli tyyppiarvoa.*

7.2 HAJONTALUVUT

Keskiluvut kuvaavat aineiston keskikohtaa. Ne eivät aina kerro aineistosta riittävästi sellaisenaan, vaan niiden tueksi tarvitaan muita lukuja. Esimerkiksi tapahtuma, jonka osallistujien ikäkeskiarvo (aritmeettinen keskiarvo) on 14 vuotta, voi yhtä hyvin olla vauvajumppa kuin yläasteikäisten tapahtuma. Yläasteen tapahtumassa kaikki osallistujat ovat noin 14-vuotiaita, kun taas vauvajumpassa puolet osanottajista on vauvoja ja puolet vanhempia. Pelkkä aritmeettinen keskiarvo ei kerro aineistosta tarpeeksi, vaan tarvitaan toinen tunnusluku kertomaan, ovatko arvot tiiviisti nipussa keskiarvon lähellä vai ovatko ne hajaantuneet laajalle alueelle. Arvojen jakautumista kuvataan hajontaluvuilla.

Hajontaluvut kertovat, millä tavalla joukon arvot ovat jakautuneet. Kun tiedetään keskiluvun lisäksi joukkoa kuvaava hajontaluku, saadaan hyvä yleiskäsitys koko tietojoukosta.

7.2.1 Vaihteluväli

Vaihteluväli on yksinkertainen keino kuvata joukon arvojen jakautumista. Kuten nimikin sanoo, vaihteluväli kertoo sen välin, minkä sisälle kaikki arvot mahtuvat. Vaihteluväli on siis väli joukon pienimmästä arvosta suurimpaan. (Henttonen ym. 2004, 310.)

7.2.2 Keskihajonta

Keskihajonta on usein käytetty tapa kuvata arvojen jakautumista. Yhdessä aritmeettisen keskiarvon kanssa se antaa hyvän käsityksen tietojoukosta.

Keskihajonta kuvaa, miten kaukana arvot keskimäärin sijaitsevat koko joukon aritmeettisesta keskiarvosta. Keskihajonta lasketaan summaamalla yhteen jokaisen arvon ja keskiarvon erotuksen neliö. Summa jaetaan arvojen lukumäärällä ja siitä otetaan neliöjuuri. (Henttonen ym. 2004, 310).

Esimerkiksi lukusarjan 0, 1, 1, 2, 3, 6, 6 keskihajonta on esitetty kuvassa 12.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} \left(x_i - \overline{x}\right)^2}{n}}$$

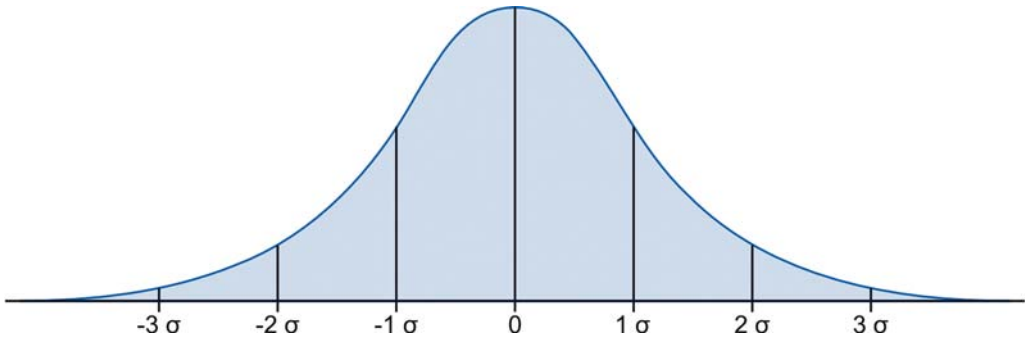
$$\sigma = \sqrt{\frac{(0-2,71)^2 + (1-2,71)^2 + (1-2,71)^2 + (2-2,71)^2 + (3-2,71)^2 + (6-2,71)^2 + (6-2,71)^2}{7}}$$

$$\sigma = 2,25$$

KUVA 12. *Aritmeettinen keskihajonta.*

7.3 NORMAALIJAKAUMA

Normaalijakauma (*normal distribution, gaussian distribution*) on yleisin käytössä olevista todennäköisyysjakaumista. Jakauma on symmetrinen ja suurin osa arvoista sijoittuu lähelle aritmeettista keskiarvoa (kuva 12). Normaalijakauman tiheysfunktio on Gaussin käyrä (kuva 13). Nimensä se on saanut nimensä saksalaismatemaatikko Friedrich Gaussilta (1777–1855). Gauss käytti tähtitieteen mittausvirheiden jakauman mallintamiseen normaalijakaumaa. (Holopainen & Pulkkinen)



KUVA 13. *Normaalijakauma.*

Normaalijakauma kytkeytyy aritmeettiseen keskiarvoon ja keskihajontaan. Normaalisti jakautuneessa joukossa 68,27 % havaintoarvoista on sijoittunut aritmeettisen keskiarvon ympärille välille -1 σ…+1 σ. Vastaavasti ±2 σ sisään mahtuu jo 95,45 % ja ±3 σ sisään 99,73 % havaintoarvoista. (Spiegel & Stephens 1999, 156–157, 522.)

8 KAREL-OHJELMOINTIKIELI

Lähteenä Sakari Koivusen opinnäytetyö.

Tyypillisesti robotin työohjelma sisältää liikekäskyjä, yksinkertaisia laskutoimituksia, tulojen ja lähtöjen käsittelyä sekä ehto-, toisto- ja hyppyrakenteita. Ohjelma voidaan tehdä robotin käsiohjaimella ja sen muokkaaminen onnistuu niin ikään paikan päällä käsiohjaimella. (Kuivanen 1999, 79–80.)

Karel on Fanuc-roboteissa käytettävä kieli, joka täydentää normaalien työohjelmien toiminnallisuutta. Sillä voidaan esimerkiksi käsitellä tiedostoja ja käyttää sarjaporttia. Karel-ohjelmia ei luoda tai muokata robotin ohjaimella, vaan ne kirjoitetaan tietokoneella. Lähdekoodimuodossa oleva Karel-tiedosto käännetään ns. p-koodiksi erillisellä kääntäjällä, jonka jälkeen p-koodi ladataan robotille suoritettavaksi. (Fanuc Robotics 2006, 45.)

Osa muistakin robottimerkeistä luottaa vastaavaan kahden ohjelmointikielen lähestymistapaan. Joillakin taas on käytössä vain yksi ohjelmointikieli, jossa vaativampia toimituksia, kuten sarjaliikenteen käsittelyä, tehdään osana normaalia työohjelmaa.

OTC- ja Nachi-roboteissa käytetään samaa AX-ohjausjärjestelmää, jossa on monimutkaisempia rakenteita varten Robot Language -ohjelmointikieli. Sen avulla voidaan käsitellä sarjaliikennettä, mutta tiedostojen käsittely sillä ei onnistu. (Puhelinkeskustelu, Niko Moilanen, 19.5.2010; sähköpostikeskustelu, Niko Moilanen, 27.5.2010).

Robot Language perustuu SLIM-kieleen (Standard Language for Industrial Manipulator) ja noudattaa JIS8439-standardia. Se muistuttaa syntaksiltaan Basic-ohjelmointikieltä. Ohjelmakoodi tehdään tekstieditorilla joko tietokoneella tai robotin ohjausyksikön ascii-editorilla. Ohjelma voidaan kääntää joko tietokoneella tai robotin ohjausyksikössä. RS232-liikenteen käsittely onnistuu Robot Languagen input- ja print-komennoilla. (Nachi Fujikoshi Corp.).

ABB-roboteissa käytetään samaa ohjelmointikieltä sekä yksinkertaisiin liikeohjelmiin että vaativampia rakenteita edellyttävään ohjelmointiin. Rapid-ohjelmointikieli tukee sekä sarjaliikenteen että tiedostojen käsittelyä. Tietoa voidaan tallentaa ascii-muotoisiin taulukkomuuttujiin, joissa tieto on pilkuilla erotetuissa kentissä. Ohjelmassa on mahdollista myös kirjoittaa suoraan tiedostoon rivi kerrallaan. (Puhelinkeskustelu, Pentti Tolonen, 25.5.2010; sähköpostikeskustelu, Pentti Tolonen, 2.6.2010.)

Motoman-roboteissa on vakiona RS-232 -tiedonsiirtoportti. Sarjaliikenteen käsittelyyn ohjaus tarjoaa host-protokollan, joka on tarkoitettu parametrityyppisten tietojen siirtoon. Sarjaliikenteen käsittely ei tapahdu varsinaisen työohjelman sisällä, vaan host-protokollan kautta tuleva data siirretään suoraan valittuihin muistirekistereihin. Protokolla on tarkoitettu yksisuuntaiseen tiedonsiirtoon, sen avulla ei ole mahdollista lähettää tietoa. Esimerkiksi mittauksen käynnistävä käsky voidaan antaa digitaalilähtönä, jonka jälkeen host-protokolla kuuntelee mittalaitteen vastauksen ja palauttaa sen muistirekisteriin. Tiedostojen käsittely ei ole mahdollista Motoman-ohjauksessa. (Sähköpostikeskustelu, Jari Kallonen, 25.5.2010; puhelinkeskustelu, Jari Kallonen, 7.6.2010.)

8.1 KIELEN SYNTAKSI

Yksinkertainen Karel-ohjelma näyttää seuraavalta:

```
PROGRAM hei
%COMMENT = 'Testiohjelma'
VAR
     luku : INTEGER
BEGIN
     -- tämä on kommentti testiohjelmassa
     luku = 4
     WRITE TPDISPLAY('Hei, maailma!', cr)
     WRITE TPDISPLAY(luku, cr)
END hei
```

Ohjelmointikielen syntaksi ei edellytä sisennystä, mutta lähdekoodin luettavuuden kannalta on suositeltavaa erottaa ohjelmalohkot sisennyksellä. Ohjelmakoodin kirjainkoolla ei ole merkitystä kuin luettavuuden kannalta – tässä työssä on käytetty samaa käytäntöä kuin Karel-referenssimanuaalissa, jossa kielen varatut sanat on kirjoitettu isolla.

Alussa määritetään ohjelman nimi `PROGRAM`-käskyllä. Ohjelman määritelty nimi voi poiketa tiedoston nimestä. Lähdekooditiedoston, käännetyn tiedoston ja ohjelman nimen ei siis tarvitse vastata toisiaan. `PROGRAM` on ohjelman ensimmäinen käsky. Sitä ennen ei voi olla muita komentoja, ainoastaan kommenttirivejä. Käskyn vastaparina on ohjelman lopussa oleva `END`, jossa annetaan ohjelman nimi samalla tavalla kuin ohjelman alussa käskyn `PROGRAM` perässä. (Fanuc Robotics 2006, A-271–A-272.)

Heti ohjelman nimen määrittelyn jälkeen voidaan antaa Karel-kääntäjälle määrityksiä. Kääntäjän määritykset merkitään prosenttimerkillä rivin alussa. Esimerkkiohjelmassa on vain yksi kääntäjän määritys, `%COMMENT`. Tällä annetaan ohjelmalle kommentti, joka näkyy robotin ohjelmaluettelossa. Ohjelman kommentin maksimipituus on 16 merkkiä. Kääntäjän määritykset pitää tehdä `PROGRAM`-käskyn jälkeen, mutta kuitenkin ennen muuttujien määrittelyä. (Fanuc Robotics 2006, 64, A-72).

Muita kääntäjän määrityksiä ovat esimerkiksi

- `%INCLUDE`, jolla voidaan sisällyttää jokin toinen tiedosto käsiteltävään lähdekooditiedostoon kääntämisen yhteydessä (Fanuc Robotics 2006, A-176 - A-177)
- `%NOLOCKGROUP`, jolla voidaan määritellä niin, että ohjelma ei varaa itselleen liikeakseliryhmiä (Fanuc Robotics 2006, A-236)
- `%ENVIRONMENT`, jolla ohjelmaan sisällytetään kääntämisen yhteydessä ympäristötiedosto. Ympäristötiedostoissa on määritelty sisäänrakennettuja funktioita, vakioita ja järjestelmämuuttujia (Fanuc Robotics 2006, A-128).

Ohjelmassa voi käyttää itse määriteltyjä muuttujia. Muuttujan nimi ei voi ylittää 12 merkkiä ja sen pitää alkaa kirjaimella. Muuttujan nimenä ei voi käyttää Karel-kielen varattuja sanoja, kuten `WRITE` tai `IF`. Muuttujat määritellään ennen varsinaisen ohjelman alkamista `VAR`-lohkossa. Jokaiselle muuttujalle pitää määrittää tyyppi. Mahdollisia muuttujatyyppejä ovat:

- Boolean, totuusarvo. Boolean-muuttuja voi saada kaksi arvoa, tosi ja epätosi. Kelvolliset arvot boolean-muuttujalle ovat TRUE tai ON (tosi) sekä FALSE tai OFF (epätosi) (Fanuc Robotics 2006, A-41.)
- File, tiedostomuuttuja. Jos ohjelmassa käsitellään tiedostoja tai tiedonsiirtoportteja, ne pitää määritellä file-tyyppisinä muuttujina. (Fanuc Robotics 2006, 7–3.)
- Integer, kokonaisluku. Kokonaislukumuuttuja voi pitää sisällään luvun välillä 2147483648...2147483646. Integer-tyypin muuttuja ei voi sisältää erikoismerkkejä tai desimaalierotinta, vaan se on aina puhdas kokonaisluku. (Fanuc Robotics 2006, A-201.)
- Real, liukuluku. Real-tyyppinen muuttuja sisältää desimaaliosan, toisin kuin integer. Muuttujan arvo voi olla -3.4028236E+38...-1.175494E-38, 0.0 tai +1.175494E-38...+3.4028236E+38. Desimaalierottimena käytetään pistettä, luvussa ei saa esiintyä pilkkua tai välilyöntiä. (Fanuc Robotics 2006, A-286.)
- String, merkkijono. Merkkijono on tekstityyppinen muuttuja, jolle annetaan muuttujien määrittelyssä maksimipituus hakasulkujen sisällä. Maksimipituus merkkijonolle on Karel-referenssimanuaalin mukaan 128, mutta ainakin robotin ohjausversio 7.40 hyväksyy pituudet 256 merkkiin saakka. (Fanuc Robotics 2006, A-338.)

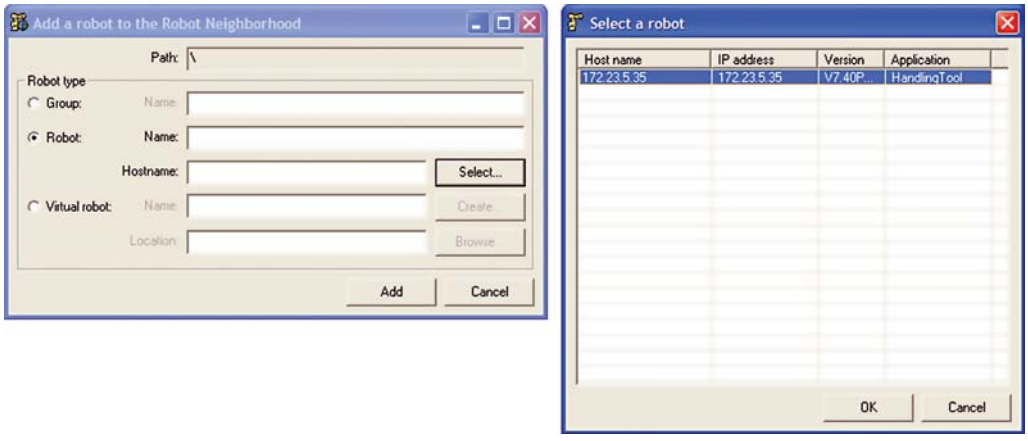
Lisäksi muuttujista voi muodostaa taulukoita (*array*). Muuttujataulukot määritellään syntaksilla `taulukko : ARRAY[4] of integer`, jolloin muodostetaan muuttujat `taulukko[1]`, `taulukko[2]`, `taulukko[3]` ja `taulukko[4]`. Yksiulotteisten ja kiinteän kokoisten taulukkojen lisäksi voidaan muodostaa moniulotteisia tai muuttuvan kokoisia taulukoita. Muuttuvan kokoisia taulukoita koskee kuitenkin rajoitus: niiden koko pitää määritellä ennen Karel-ohjelman suorittamista. Taulukon koon muuttaminen ei siis onnistu ohjelman suorittamisen aikana. (Fanuc Robotics 2006, 2-18–2-20.)

8.2 OHJELMAN KÄÄNTÄMINEN JA LATAAMINEN ROBOTILLE

Kääntäjäksi sanotaan ohjelmaa, joka ohjelmointikielellä kirjoitetun lähdekoodin perusteella luo suoritettavan binääritiedoston. Karel-lähdekoodi pitää ensin kääntää niin sanotuksi p-koodiksi tietokoneella. Sen jälkeen se voidaan ladata robotin ohjaimelle suoritettavaksi. Kääntäjä ilmoittaa mahdollisista syntaksivirheistä ja tuottaa p-kooditietoston vain, jos lähdekoodi on syntaktisesti oikein. Käännettyjen tiedostojen tunniste on .pc. (Fanuc Robotics 2006, 1–3.)

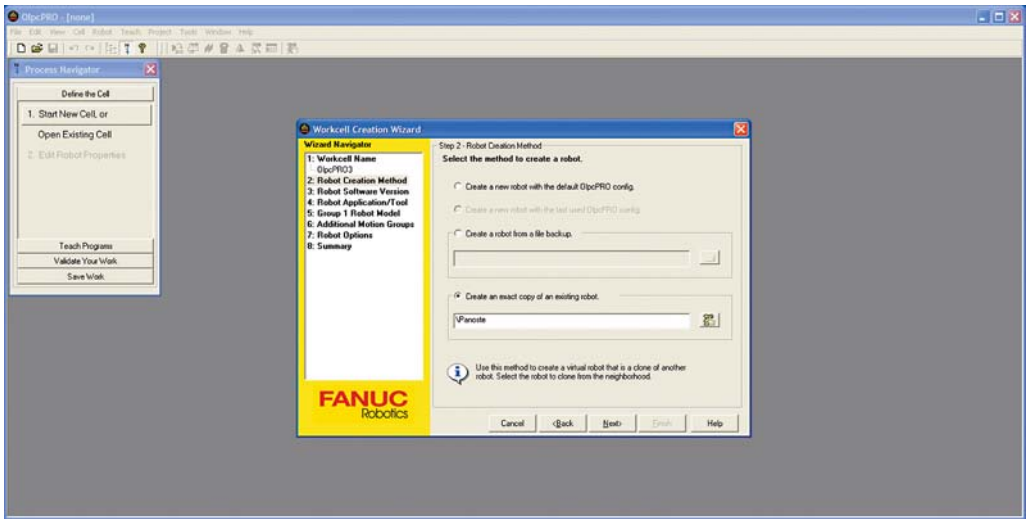
Karel-kääntäjä tulee OlpcPRO -ohjelmapaketin mukana. OlpcPRO on yksinkertainen offline-ohjelma, jossa ei ole 3D-simulointiympäristöä tai sovelluskohtaisia toimintoja esimerkiksi hitsaukseen tai maalaamiseen. Se on tarkoitettu kevyeksi vaihtoehdoksi Fanucin varsinaiselle Roboguide-simulointiympäristölle.

Ohjelmaan on mahdollista tuoda olemassa olevan robotin varmuuskopio. Varmuuskopion tiedostoja voidaan selata ja muokata tietokoneella, ja robotin asetuksiin voidaan tehdä muutoksia. Tiedostot voidaan siirtää takaisin robotille esimerkiksi USB-tikun avulla.



KUVA 14. *Robot neighborhood.*

R-30iA-ohjauksessa on vakiona Ethernet-liitäntä (Fanuc Robotics b, 380). Jos tietokone ja robotti on liitetty samaan verkkoon, voidaan OlpcPROlla ottaa yhteys robottiin suoraan verkon kautta. Kuvassa 14 näkyvässä robot neighborhood -valikossa voidaan lisätä verkossa olevat robotit OlpcPROn näkymään. Tässä robotille on annettu IP-osoite 172.23.5.35.



KUVA 15. *Uuden robottisolun luominen OlpcPRO-ohjelmassa.*

Kun verkossa oleva robotti on tuotu ohjelmaan ja sille on annettu nimi, voidaan sen asetukset ja tiedostot tuoda OlpcPROn käyttöön. Ohjelmassa luodaan uusi robottisolu joko puhtaalta pöydältä, olemassa olevan solun varmuuskopiosta tai olemassa olevasta solusta verkon kautta. Kuvassa 17 näkyy, miten verkossa olevan robotin tiedot tuodaan ohjelmaan.

Työn alkuvaiheessa käytössä ollut OlpcPRO-versio ei ollut yhteensopiva robotin ohjausversion kanssa, minkä vuoksi robottiin ei voitu ottaa suoraan yhteyttä OlpcPROsta käsin. OlpcPROn virtuaalirobotti piti päivittää tukemaan ohjausversiota 7.40. Karel-ohjelmia voidaan kääntää ilman OlpcPRO-versiotakin.

Karel-kääntäjä sijaitsee OlpcPRO-asennushakemiston alla bin-alihakemistossa nimellä ktrans.exe. Karel-lähdekoodi voidaan kääntää kahdella eri tavalla: joko OlpcPRO-ohjelman kautta tai suoraan kääntäjää käyttäen. Koska OlpcPRO ei versioristiriidan vuoksi toiminut alussa, käytettiin Karel-kääntäjää suoraan. Tekstieditorilla kirjoitettu Karel-lähdekoodi voidaan kääntää komentorivillä yksinkertaisesti syntaksilla `ktrans.exe lähdekoodi.kl`. Lisämääreeksi ohjelmalle voidaan antaa kohdetiedoston nimi ja robotin ohjauksen versio, jolle ohjelma käännetään.

Lähdekoodi kirjoitettiin Gnome-työpöytäympäristön Gedit-tekstieditorilla, joka on saatavissa myös Windows-käyttöjärjestelmälle. Linuxissa Gedit voi suo-

rittaa käyttäjän määrittelemiä ulkoisia komentoja. Tätä voi käyttää esimerkiksi kääntäjän käynnistämiseen niin, että pikanäppäin tallentaa työn alla olevan lähdekoodin, ajaa sen kääntäjän läpi ja ilmoittaa editori-ikkunan alalaidassa kääntämisen onnistumisesta tai kääntäjän palauttamista virheistä. Koska Windows-käyttöjärjestelmästä puuttuu Unix-sukuisten järjestelmien tehokas komentorivi, tätä ominaisuutta ei voi toistaiseksi käyttää Windowsissa (Gnome bugzilla 2010).

Kääntämiseen tehtiin kiertoratkaisu, joka helpotti työn tekemistä. Yksinkertainen komentojonotiedosto karel-kääntö.bat sisältää kaksi riviä:

```
"C:\Program files\Fanuc\WinOLPC\bin\ktrans.exe" /ver V7.40-1
%1 "%~dpn1.pc"
pause
```

Kun näytöllä Karel-tiedosto raahataan tämän komentojonotiedoston päälle, ohjelma ajetaan Karel-kääntäjän läpi ja kääntäjän antama tuloste näkyy ruudulla, kunnes käyttäjä painaa jotain näppäintä. Tämä osoittautui melko käteväksi tavaksi kääntää Karel-koodia.

OlpcPRO-ympäristössä on mahdollista testata Karel-ohjelmien toimintaa. Sarjaportin käsittely ei kuitenkaan onnistu virtuaalirobotilla, joten ohjelmia testattiin oikealla robotilla virtuaaliympäristön sijaan.

8.3 TYÖSSÄ KÄYTETTYJÄ KAREL-KÄSKYJÄ

Ohjelmien ymmärtämisen helpottamiseksi tässä käydään läpi joitakin usein esiintyviä Karel-ohjelmointikielen käskyjä.

8.3.1 Tiedostojen ja tiedonsiirtoporttien käsittely

Tiedostojärjestelmän ja tiedonsiirtoporttien käyttäminen onnistuu Karelissa file-muuttujatyypin avulla. Tiedonsiirtoportteja käsitellään siis kuten tiedostoja. Tiedostojen käsittelyyn on neljä perustoimintoa: tiedoston avaaminen, tiedostoon kirjoittaminen, tiedostosta lukeminen ja tiedoston sulkeminen.

`OPEN FILE` avaa tiedoston tai portin ja liittää sen haluttuun muuttujaan. Komennolle annetaan kaksi parametria. Ensimmäinen parametri kertoo tiedoston käyttötavan ja toinen tiedoston sijainnin ja nimen tai tiedonsiirtoportin tunnuksen. (Fanuc Robotics 2006, 7–4.)

Esimerkiksi `OPEN FILE lokitiedosto('AP','UD1:loki.txt')` avaa USB-tikulla olevan tiedoston loki.txt append-muodossa ja antaa tälle tiedostomuuttujalle nimeksi lokitiedosto. Muuttujan nimellä viitataan avattuun tiedostoon myöhemmin ohjelmassa.

Käyttötapa kertoo, miten tiedostoa käytetään. Vaihtoehtoja ovat:

- RO (read only): Vain lukutoimintoja varten. Asettaa lukuosoittimen tiedoston alkuun. Tiedoston pitää olla olemassa. (Fanuc Robotics 2006, 7–11.)
- RW (read/write): Sekä luku- että kirjoitustoimintoja varten. Kirjoittaa vanhan tiedon päälle. Asettaa osoittimen tiedoston alkuun. Jos tiedostoa ei löydy, se luodaan. (Fanuc Robotics 2006, 7–12.)
- AP (append): Lisää olemassa oleviin tietoihin. Sopii sekä lukemiseen että kirjoittamiseen, mutta ensimmäisen toiminnon pitää olla kirjoitus. Asettaa osoittimen tiedoston loppuun. Jos tiedostoa ei löydy, se luodaan. (Fanuc Robotics 2006, 7–12.)
- UD (update): Tietojen päivitys. Asettaa osoittimen tiedoston alkuun. Korvaa vanhat tiedot merkki merkiltä. (Fanuc Robotics 2006, 7–12.)

Muuttuja pysyy liitettynä tiedostoon tai porttiin kunnes tiedosto suljetaan `CLOSE FILE` -käskyllä tai ohjelman suorittaminen päättyy.

`WRITE`-komennolla voidaan kirjoittaa joko tiedostoon, tiedonsiirtoporttiin tai näytölle. Kirjoitettava sisältö voi olla mitä tahansa: tekstiä, lukuja, muuttujan arvo jne. Esimerkiksi `WRITE TPDISPLAY('Tämä on testi', cr)` tulostaa kohteeseen `TPDISPLAY` tekstin ”Tämä on testi” ja rivinvaihdon (cr). `TPDISPLAY` on robotin ohjausyksikön user-näkymän tekstiruutu. Vastaavasti tiedostoon voidaan kirjoittaa komennolla `WRITE LOKITIEDOSTO('Tämä on testi', cr)`, jolloin lokitiedosto-nimellä avattuun tiedostoon kirjoitetaan tekstiä, joka päättyy rivinvaihtoon.

Sarjaporttiin kirjoittaminen on yhtä helppoa kuin tiedostoon kirjoittaminen. Sarjaportista lukemiseen kuitenkin liittyy tiettyjä ominaispiirteitä. Sarjaporttiin tuleva data kootaan tiedonsiirtopuskuriin, josta tietoa voidaan lukea haluttu määrä. Puskurin pituus on 256 tavua. (Fanuc Robotics 2006, 7–17.)

Puskurista pitää lukea dataa tietty määrä kerrallaan. Jos yritetään lukea enemmän merkkejä kuin puskurissa on, jää ohjelma odottamaan puuttuvia merkkejä loputtomiin. On siis tarpeen rajata luettava määrä. Tämä tapahtuu kahden muotomäärityksen avulla, jotka erotellaan muuttujasta ja toisistaan kahdella kaksoispisteellä, esimerkiksi `muuttuja::n::m`, missä n ja m ovat kokonaislukuja. (Fanuc Robotics 2006, 7–18.)

Eri muuttujatyypeillä on hieman toisistaan poikkeavat muotoilukäytännöt, mutta kaikilla muuttujatyypeillä ensimmäinen määritys kertoo, miten monta merkkiä luetaan. Liukulukujen (real) ja totuusarvomuuttujien (Boolean) tapauksessa toisella määrityksellä ei ole merkitystä. Kokonaislukumuuttujaa (integer) luettaessa toinen määritys kertoo käytössä olevan lukujärjestelmän kantaluvun välillä 2...16. Tekstimuuttujaa (string) luettaessa toisella määrityksellä ilmoitetaan, onko tieto kääritty lainausmerkkeihin vai ei. (Fanuc Robotics 2006, 7–30.)

8.3.2 Ehtorakenteet

Ohjelmissa vaaditaan usein toimintoja, jotka suoritetaan mikäli tietty ehto toteutuu. Esimerkki ehtorakenteesta voisi olla, mittatuloksen osumisen varmistamisesta toleranssialueelle. Jos mitta on oikealla alueella, suoritetaan toiminto 1. Jos mitta ei osu alueelle, suoritetaan toiminto 2.

Yleinen ehtorakenne on `IF…ENDIF`. Sen syntaksi on yksinkertainen:

```
IF ehto THEN
    koodia, joka ajetaan kun ehto toteutuu
ELSE
    koodia, joka ajetaan kun ehto ei toteudu
ENDIF
```

Else-lohko on valinnainen ja sen tarpeellisuus on tapauskohtaista. (Fanuc Robotics 2006, A-174 - A-175.)

Jos ehtoja on useita, ne pitää sijoittaa sulkumerkkeihin. Esimerkiksi `IF ((luku=1) OR (luku=6))`

`SELECT` on toinen käytössä olevista ehtorakenteista. Sillä voidaan kokonaislukuarvon perusteella suorittaa yksi useista eri lohkoista. (Fanuc Robotics 2006, A-301 - A-302.)

```
SELECT luku OF
CASE(1):
    jos luku=1, suoritetaan tämä lohko
CASE(2):
    jos luku=2, suoritetaan tämä lohko
ELSE:
    jos luku ei ole 1 tai 2, suoritetaan tämä lohko
ENDSELECT
```

8.3.3 Toistorakenteet

Toisinaan ohjelmassa pitää suorittaa tietty toiminto useita kertoja peräkkäin. Tähän Karel tarjoaa työkaluksi erilaisia toistorakenteita.

FOR on usein hyödyllinen toistorakenne, kun esimerkiksi käydään läpi muuttujamatriisia. Tässä toistorakenteessa silmukka suoritetaan n kertaa kokonaislukulaskurin avulla. (Fanuc Robotics 2006, A-135 - A-136).

```
FOR luku = 1 to 10 DO
    suoritetaan kymmenen kertaa, luku kasvaa joka kerta
    yhdellä
ENDFOR
```

REPEAT…UNTIL on silmukka, jota toistetaan, kunnes ehto toteutuu. Ehto voi olla mikä tahansa lauseke, joka palauttaa Boolen totuusarvon. (Fanuc Robotics 2006, A-292 – A-293.)

```
REPEAT
    i = i+1
UNTIL i > 10
```

WHILE toimii kuten REPEAT…UNTIL, mutta päinvastaisesti. Siinä silmukkaa toistetaan niin kauan kuin ehto on voimassa. (Fanuc Robotics 2006, A-292 - A-293).

```
WHILE i < 5
    i = i+1
ENDWHILE
```

8.3.4 Muita käskyjä ja rakenteita

GET_TPE_PRM-komennolla voidaan lukea ohjelmakutsussa annetut parametritiedot muuttujiin. Ohjelma voidaan kutsua esimerkiksi käskyllä `CALL TESTI('kokeilu')`, jolloin käyttäjän syöttämä teksti 'kokeilu' voidaan TESTI-ohjelmassa lukea muuttujan sisällöksi. Ohjelmakutsussa parametri voi olla mitä tahansa tietotyyppiä, esimerkiksi tilanteessa `CALL TESTI(3.1415)` parametri sisältää liukuluvun (real).

GET_TPE_PRM-komennon syntaksi on `GET_TPE_PRM(nro, datatyyppi, prm_int, prm_real, prm_string, status)`, missä

- Nro kertoo, mikä parametri luetaan. Kun esimerkissä on vain yksi parametri, luetaan luonnollisesti parametri numero 1.
- Toinen GET_TPE_PRM-parametri palauttaa tiedon tyypin. Kun ohjelmakutsussa annetaan parametrina liukuluku, muuttujan datatyyppi sisältö kertoo, että parametrina saatiin real-tyypin muuttuja. Datatyypin mahdollisia arvoja ovat 1 (kokonaisluku, integer), 2 (liukuluku, real) sekä 3 (tekstimuuttuja, string).

- Kolmas parametri, joka esimerkissä on nimetty prm_int palauttaa kokonaislukumuuttujan sisällön. Jos ohjelmakutsun parametri on kokonaisluku, sen arvo tallennetaan tähän muuttujaan. Seuraavat kaksi parametria toteuttavat saman asian liukuluku- ja tekstitietotyypeille.
- Viimeinen parametri kertoo toimituksen onnistumisesta. Jos status on erisuuri kuin 0, on tapahtunut poikkeus. Esimerkiksi status 17042 tarkoittaa, että parametria tällä numerolla ei ole olemassa.

Robotin muistirekistereitä voidaan lukea ja kirjoittaa Karel-ohjelmassa. Rekistereitä voi käyttää esimerkiksi ohjelmien väliseen kättelyyn tai tietojen palauttamiseen Karel-ohjelmasta työohjelman käyttöön.

Liukuluvun voi sijoittaa rekisteriin käskyllä `SET_REAL_REG(rekisteri, arvo, status)`, missä rekisteri on kokonaisluku ja arvo on liukuluku. Statusmuuttujaan sijoitetaan tieto operaation onnistumisesta.

Kokonaisluku voidaan kirjoittaa rekisteriin vastaavalla toiminnolla `SET_INT_REG(rekisteri, arvo, status)`.

9 MEKAANINEN MITTALAITE PYÖRÄHDYSKAPPALEIDEN SISÄHALKAISIJOILLE

Lähteinä Sakari Koivusen opinnäytetyö, Jaakko Lennon, Juuso Raidan ja Tommi Unkurin projektiraportti sekä Dmitri Glouchenkon ja Marko Piiran projektiraportti.

Sisähalkaisijoiden mittaamiseen käytössä on Mitutoyon sisämittauslaite 511–132, johon on liitetty mittakello ID-C112GB. Mittakellon erottelukyvyksi on ilmoitettu 0,001 mm ja tarkkuudeksi 0,003 mm. Mittausliike on 12,7 mm ja näytteenottotaajuus 50 mittausta sekunnissa. (Mitutoyo, viitattu 28.5.2010.)

KUVA 16. *Sisämittauslaitteen mittauspää.*

Sisämittauslaite tukeutuu mitattavan reiän reunoihin kolmesta pisteestä, jolloin mittauskohta keskitetään mekaanisesti oikeaan paikkaan. Mittapää on esitetty kuvassa 16. Tyypillisesti sisämittauslaite on tarkoitettu manuaaliseen käyttöön, jolloin se viedään mitattavaan reikään, tehdään keinuttava mittaliike ja luetaan pienin mittaustulos. Tässä mittalaite on kuitenkin asennettu kiinteälle jalustalle kahteen suuntaan joustavan nivelen päälle niin, että robotti voi suorittaa mittauksen. Jalustaan kiinnitetty mittalaite on esitetty kuvassa 17. Mittaaminen suoritetaan viemällä kappale mittalaitteelle ja tekemällä mittausliike robotilla. Mittauksen luonteesta johtuen kosinivirhe on hyvin suuri mittausliikkeen alku- ja loppupäässä. Todellinen sisähalkaisija on mittausliikkeen keskivaiheilla. Mittakellossa on hold-toiminto, joka jättää pienimmän mitta-arvon käyttäjän nähtäville automaattisesti.

KUVA 17. *Mitutoyo-sisämittauslaite.*

Panoste-soluun suunniteltiin kiinteä mekaaninen mittapiste sorvattujen kappaleiden sisähalkaisijoiden mitoitukseen. Mittalaitteena toimii Mitutoyon reikäindikaattori. Itse mittausliike tehdään robotilla ja reikäindikaattori pysyy kiinni laitteessa.

9.1 LAITTEISTON SUUNNITTELU

Laitteistoa suunnitellessa olennaista oli, että indikaattori palaisi joka mittauksen jälkeen takaisin lähtöpaikkaansa, ja että robotin tekemä mittausliike saatiin tehtyä mahdollisimman helposti. Niinpä päädyttiin tekemään pyörivälle akselille nivel, johon mittalaite kiinnitettiin. Joka mittauksen jälkeen laite lasketaan "Y-haaran" varaan, jolloin varmistetaan, että alkupiste on aina sama. Joustoa ei laiteessa tarvita.

Laitteisto valmistettiin alumiinista koneistamalla. Mittavarren on mahdollista pyöriä kiinteään pylvääseen pultatun teflon-holkin päällä. Holkki on pultattu alumiinielementeistä valmistettuun jalustaan. Solua laajennettiin projektin aikana.. Nivelen akselina on pultti, ja mittalaite on siis helposti irrotettavissa mittatyökalujen vaihtoa varten, asemoinnin muuttumatta.

Mittalaitteen päässä on Mitutoyon mittakello, joka ilmoittaa mittausarvon tuhannesosamillin tarkkuudella. Tarkoituksen oli saada tämä tieto robotille, joka saadessaan tiedon ilmoittaa mitan esim. merkkauslaitteelle tai siirtää tavaran valmiiden pinoon tms. Mitutoyolla on oma tiedonsiirtokaapeli, jolla tieto saadaan suoraan robotille.

Ryhmä tutki lineaariantureita, jotka sopisivat tähän sovellutukseen. Anturin tuli ilmoittaa analogista tietoa 12V jännitteellä ja mittausalueen tuli olla 0 5mm. Tässä pari esimerkkiä laitteistoon mahdollisesti käyvistä antureista:

Novotechnik TR/TRB (IP/40)
Novotechnik F20U (IP65)

Mitutoyon Digimatic-kaapelia ei saatu luonnollisesti liitettyä suoraan tietokoneeseen, joten hankintalistalle tulivat DMX-1-adapteri ja 10m RS232-kaapelia.

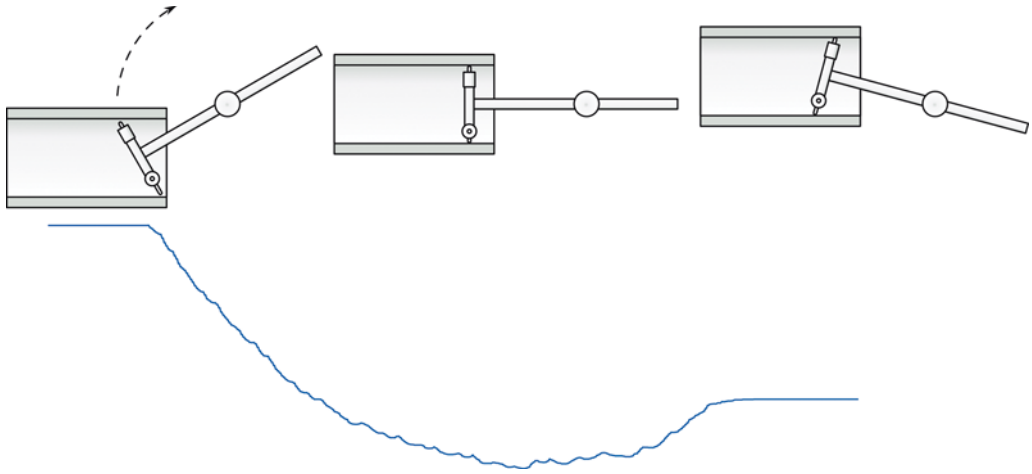
Tiedonsiirtoa alettiin heti kokeilla, kun kaapelit olivat saapuneet. Ryhmä löysi vain pienen pätkän Mitutoyon protokollaa. Sen avulla onnistuttiin lähettämään tietokoneella käsky, jolla mittalaite lähettää mitatun arvon tietokoneelle. Mittakello toimii kuitenkin niin, että joka mittauksen jälkeen se tulee nollata painamalla START-painiketta. Nollaukseen vaadittavaa koodia ei onnistuttu kuitenkaan löytämään. Asiaa kysyttiin toimittajalta, jolta tuotteet oli tilattu, mutta sieltäkään ei saatu laajempaa protokollaa aiheesta. Toimittaja kehotti hankkimaan erillisen laskurin robotin ja mittalaitteen välille. Tämän vuoksi kelloa pitää käyttää ilman nollauspainikkeen käyttöä ja solun Fanuc-robotti tekee täten mittakellon työn. Eli robotille annetaan kaikki mittauksesta tulevat arvot ja se löytää itse pienimmän mitatun arvon ja laskee oikean mitan tästä.

Mittauksessa huomioonotettuja asioita:

- epäsymmetriset kappaleet
- olakkeet
- reiät
- pinnankarheuden vaikutus mittaustarkkuuteen mittausnopeuden kasvaessa.

9.2 TIEDONSIIRTO MITTALAITTEEN JA ROBOTIN VÄLILLÄ, MITUTOYO

Mitutoyon sisämittauslaitteella mitan lukemisessa hyödynnetään yksinkertaista keinoa korjata kosinivirhe: tehdään hyvin suuri määrä mittauksia liikkeen aikana ja palautetaan mitta-arvoksi pienin liikkeen aikana luettu mittaustulos. Käsin tapahtuvassa mittauksessa mittapäätä keinutetaan kappaleen sisällä, jolloin pyyhkäisevän mittausliikkeen aikana käydään nopeasti asennossa, missä mittalaite on täysin suorassa mitattavaan kappaleeseen nähden. Tässä tapauksessa robotti liikuttaa kappaletta ja itse mittalaite on kiinnitetty nivellettyyn telineeseen. Kuvassa 18 on esitetty mittauksen toimintaperiaate.



KUVA 18. *Mittausliike ja mitta-arvo liikkeen aikana.*

9.2.1 Vaatimusmäärittely

Yksittäisen mitan lukeminen ei ole tarpeellista Mitutoyon tapauksessa, koska kaikki tällä kokoonpanolla tehtävät mittaukset ovat kuvan 18 mukaisia pyyhkäiseviä mittauksia. Rajapinnan pitää siis toteuttaa vain kaksi asiaa mittatietojen lukemisen suhteen: liikkeen aikaiset kättelysignaalit rajapinnan ja työohjelman välillä sekä liikkeen aikaisen minimimitan löytäminen.

Poikkeusten käsittelyä koskevat samat vaatimukset kuin Keyencen optista mikrometriä.

Mitutoyon mittalaitteessa ei ole asetuksia, joita pitäisi muuttaa robottiohjelmasta käsin.

9.2.2 Toteutus

Mitutoyon tiedonsiirtoprotokolla on hyvin suoraviivainen. Mittatiedot luetaan lähettämällä sarjaporttiin luku 1. Viesti päätetään CR-rivinvaihdolla. Mittalaite vastaa tässä tapauksessa aina muodossa 01A+xxxx.xxx ja päättää viestin CR-rivinvaihtoon. Alun neljä merkkiä kertovat mittalaitteen ja kanavan lisäksi mittaustuloksen etumerkin. Näiden jälkeen seuraavat kahdeksan merkkiä ovat mittadataa, jossa desimaalierottimena käytetään pistettä.

Jos mittalaite ei ole päällä, se palauttaa DMX-1 virhekoodin. Virhekoodissa on kolme merkkiä, joista ensimmäinen on aina 9. Jälkimmäiset kertovat mittalaitteen kanavatunnuksen ja virheen tyypin. Esimerkiksi 911 kertoo mittalaitteen olevan pois päältä.

9.2.3 Suunnitteludokumentti

```
MITUTOYO(portti, tulosrekisteri, kättelyrekisteri)

portti:

    - 1: portti ohjauskaapin etupaneelissa oleva liitin

    - 2: portti ohjauskaapin sisällä oleva liitin
         tulosrekisteri:

    - rekisteri, johon tulos palautetaan mittauksen
         päätyttyä

kättelyrekisteri:

    - rekisteriä käytetään interfacen ja robotin liikeoh-
jelman väliseen kättelyyn

    - 0: alku

    - 1: robotille tieto -> saa aloittaa liikkeen

    - 2: tieto robotilta -> liike tehty

    - 3: tieto robotille -> mittatulos luettu muistiin,
         saa poistua

pseudokoodi
```

```
lue parametri 1 (portti)
    jos ei löydy, palauta virhe
    jos on väärää tyyppiä (ei kokonaisluku), palauta virhe
    tallenna oikean arvon perusteella portin tunnus portti-
    muuttujaan

lue parametri 2 (tulosrekisteri)
    jos ei löydy, palauta virhe
    jos on väärää tyyppiä (ei kokonaisluku), palauta virhe
    tallenna oikea arvo rekisteri-muuttujaan

lue parametri 3 (kättelyrekisteri)
    jos ei löydy, palauta virhe
    jos on väärää tyyppiä (ei kokonaisluku), palauta virhe
    tallenna oikea arvo kattelyrek-muuttujaan

nollaa kättelyrekisteri varmuuden vuoksi
avaa sarjaportti
lue ensimmäinen mittaustulos ja aseta se minimiksi
anna robotille liikelupa (kattelyrek=1)
toista, kunnes kattelyrek=2
    lue mittaustulos
    jos tulos<edellinen minimi, aseta tulos uudeksi
    minimiarvoksi
aseta minimiarvo tulosrekisteriin
anna robotille lupa lähteä (kattelyrek=3)
sulje sarjaportti
```

9.2.4 Ongelmia tiedonsiirrossa

Protokollan yksinkertaisuudesta huolimatta rajapinnan tekemisessä ja testaamisessa ilmeni ongelmia. Mittalukema pyydetään mittalaitteelta lähettämällä väylään merkki 1 ja rivinvaihto. Tätä testattiin tietokoneella Realterm-terminaaliohjelman avulla ja kommunikointi toimi moitteettomasti. Kun robotin ja mittalaitteen rajapinnan ensimmäinen versio saatiin testivaiheeseen, se ei toiminut odotetusti vaan jäi jumiin. Jumiutuminen johtui siitä, että robotin ohjelma jäi lukemaan sarjaportin puskurista mittatietoa, mutta koska puskurissa ei ollut mitään luettavaa, ohjelma jäi odottamaan.

Ohjelmaan tehtiin muutos virheen paikallistamista varten. Ensimmäisessä versiossa tieto luettiin kahdessa osassa: ensin neljä merkkiä, joiden perusteella päätellään tiedonsiirron eheys ja lopuksi yhdeksän merkkiä, jotka muodostuvat varsinaisesta mittatiedosta ja tiedonsiirron päättävästä rivinvaihdosta. Alussa luettavat neljä merkkiä muutettiin luettavaksi yksi kerrallaan ja välittömästi lukemisen jälkeen merkki tulostetaan näytölle.

```
READ sarjaportti(mittaustulos::1::0)
WRITE TPDISPLAY(mittaustulos)
READ sarjaportti(mittaustulos::1::0)
WRITE TPDISPLAY(mittaustulos)
READ sarjaportti(mittaustulos::1::0)
WRITE TPDISPLAY(mittaustulos)
READ sarjaportti(mittaustulos::1::0)
WRITE TPDISPLAY(mittaustulos)
```

Mitään ei kuitenkaan tulostunut, sarjaliikennepuskuri oli siis tyhjä. Tähän looginen syy on se, että mittalaitteelle lähetetty käskyviesti ei ole mennyt perille. Ensimmäinen epäily oli, että robotilta lähetetty käskyn päättävä rivinvaihto on väärä. Viestin oikeellisuutta testattiin kytkemällä sarjakaapeli robotin ja tietokoneen välille. Liikennettä seurattiin Realtermillä. Kun Mitutoyo-kommunikointi käynnistettiin robotilta, Realtermiin kirjautui aivan oikein muotoiltu viesti. Asia varmistettiin vielä tutkimalla viestiliikennettä heksakoodimuodossa. Robotilta tuleva viesti varmistui oikeaksi, heksamuodossa 31 0D eli numero 1 ja rivinvaihto.

Seuraavaksi testattiin uudelleen tietokoneen ja Mitutoyon mittalaitteen välistä kommunikointia. Mittalaitteelle lähetettiin sama koodi, joka oli saatu robotilta. Koodi lähetettiin heksamuodossa, jotta voidaan olla varmoja, että lähetetty viesti on täysin sama kuin robotin lähettämä. Mittalaite vastasi normaalisti ja toimi odotetusti. Liikenne toimi siis sekä tietokoneen ja robotin että tietokoneen ja mittalaitteen välillä täysin oikein. Sen sijaan mittalaitteen ja robotin välinen tiedonsiirto ei toiminut.

Kun vikaa etsittiin, huomattiin Realterm-ohjelman avulla, että tietokoneen ja robotin välisessä kommunikoinnissa ohjaussignaalit CTS, DCD ja DSR ovat kytkettyinä, kun taas tietokoneen ja mittalaitteen välisessä kommunikoinnissa nämä signaalit eivät ole käytössä.

Mitutoyon mittalaitteisiin saa erilaisia tiedonsiirtokaapeleita. Eräässä mallissa on mittalaitteen päässä data-painike, joka lähettää mittalaitteen lukeman sarjaväylää pitkin eteenpäin. Tämä vastaa samaa kuin väylän kautta lähetetty mittatietokutsu. Koska eräässä toisessa mittalaitteessa olevassa kaapelissa oli data-painike tiedon lähettämistä varten, kokeiltiin vikaa paikallistaa eri kaapelilla. Koska hypoteesi

oli, että robotin lähettämä ohjaussignaali ei tule perille mittalaitteelle, päätettiin ohittaa ohjaussignaalin lähettäminen ja korvata se manuaalisesti data-painikkeella. Testaaminen onnistui ja mittalaite lähetti tuloksen ongelmitta robotille.

Näillä yksinkertaisilla testeillä saatiin rajattua virhelähde. Ongelman oli pakko olla siinä, että robotti ei syystä tai toisesta lähetä viestiä mittalaitteelle. Mitutoyo DMX-1:n teknisissä määrittelyissä mainitaan laitteen kytketyt signaalijohtimet. Esimerkiksi clear to send -signaalia ei ole kytketty. Tämä kertoo lähettävälle laitteelle, että vastaanottaja on valmis käsittelemään tietoa. Mittalaitteen ja robotin väliin tehtiin uusi kaapeli, jonka vuonvalvonta ja kättely johdotettiin kuten nollamodeemikaapeli, mutta signaalijohtimet johdotettiin suoraan. Kaapeli on suora siksi, että mittalaitteen ja PC:n väliin tarkoitettu erikoiskaapeli on jo valmiiksi käännetty ja tämä suora välikaapeli tuli edellisen kaapelin jatkeeksi.

9.2.5 Ohjelman toteutus

Kun mittalaitteen ja robotin välille saatiin toimiva tiedonsiirtokaapeli, alkoi itse ohjelman kehittäminen ja testaaminen suunnitteludokumentin mukaan.

Mitutoyon mittakellossa on sisäänrakennettu hold-toiminto, joka jättää pienimmän mittauksen aikaisen lukeman mittakellon näyttöön. Tätä toimintoa voidaan hyödyntää manuaalisessa käytössä. Sitä ei kuitenkaan ole mahdollista käyttää sarjaportin kautta, koska edellisen mittauksen nollaavaa start-käskyä ei ole mahdollista lähettää sarjaväylän yli (Sähköpostikeskustelu, Harri Salmi, 28.5.2010).

Koska mittalaitteen omaa alamitan pitotoimintoa ei ole mahdollista käyttää, pitää sama toiminnallisuus toteuttaa robotilla ohjelmallisesti. Tämä on sinänsä varsin suoraviivaista: tehdään monta mittausta ja tallennetaan pienin mittausarvo muuttujaan minimi. Jos uusi mittaustulos on pienempi kuin minimi, asetetaan se minimin uudeksi arvoksi

Kun ohjelma alkaa, avataan sarjaportti ja luetaan ensimmäinen mittaustulos. Tämä tallennetaan muuttujan minimiarvoksi. Tiedonsiirron eheyttä testataan lukemalla viestipaketin kolme ensimmäistä merkkiä. Protokollan teknisen kuvauksen mukaan onnistuneessa tiedonsiirrossa kolme ensimmäistä merkkiä ovat 01A. Jos mittakello on pois päältä, kolmimerkkinen vikakoodi on 911.

Kun ensimmäinen mittaus on suoritettu onnistuneesti, aloitetaan varsinainen mittaussilmukka, jossa luetaan mittausdata ja verrataan sitä edellisiin tuloksiin. Ohjelman alussa on tallennettu lähtöarvo muuttujaan minimi. Mittaussilmukassa jokaista mittaustulosta verrataan tähän muuttujaan. Jos mittaustulos on pienempi kuin minimi, se asetetaan minimin uudeksi arvoksi. Näin pienin mittausliikkeen aikana luettu mittatulos jää muuttujaan minimi.

Rajapinnan ja robotin työohjelman välinen kättelysignaalien vaihto toteutetaan kättelyrekisterin avulla. Käyttäjä voi ohjelmakutsussa määritellä rekisterin, jota käytetään kättelyyn. Tarvittavat kättelytiedot:

- Ohjelma alkaa välittömästi ohjelmakutsun jälkeen.
- Kun rajapinta on lukenut ensimmäisen mittaustuloksen (muuttujaan minimi) onnistuneesti, se antaa robotin työohjelmalle kättelysignaalin 1, jonka jälkeen robotti aloittaa mittausliikkeen.
- Kun robotin mittausliike on valmis, annetaan rajapinnalle kättelysignaali 2 ja jäädään odottamaan viimeistä kättelytietoa ennen kuin lähdetään robotilla pois mittalaitteen luota.
- Kun rajapinta on saanut viimeisen kättelytiedon (3), se siirtää minimilukeman tulosrekisteriin. Vasta kun rekisteriin on kirjoitettu onnistuneesti, annetaan robotille lupa lähteä pois.

9.2.6 Tarkkuuden ja nopeuden suhde

Mitutoyon mittakellon ID-C112GB näytteenottotaajuudeksi on ilmoitettu 50 mittausta sekunnissa (Mitutoyo, viitattu 28.5.2010). Tämä rajoittaa mittausliikkeen nopeutta. Maksiminopeutta ei kuitenkaan voi arvioida pelkän mittakellon näytteenottotaajuuden perusteella, koska asiaan vaikuttaa lisäksi sarjaliikenteen tiedonsiirtonopeus ja tiedon käsittely sekä mittalaitteella että robotin ohjauksessa. Tiedonsiirron viemä aika voidaan laskea siirrettävän datamäärän ja tiedonsiirtonopeuden perusteella, mutta se ei kerro vielä koko totuutta. Totuudenmukaiseen nopeuden arvoon päästään kiinni ainoastaan testaamalla suorituskykyä käytännössä.

Mittauksen nopeutta testattiin yksinkertaisella ohjelmalla, joka juoksuttaa mittaussilmukassa millisekuntiajastinta ja palauttaa mittaussyklin kestoajan. Näin tehdyissä kokeissa todettiin yhden mittaussyklin vievän aikaa noin 65 ms. Nopeus on siis jonkin verran mittalaitteen näytteenottokykyä hitaampi.

Nopeassa liikkeessä mittalaite ei ehdi rekisteröidä minimimittaa oikein ja näin halkaisija tulkitaan liian suureksi. Tähän yksinkertaisin ratkaisu on se, että mittausliike suoritetaan hitaasti. Jos mittaukseen käytettävä aika on hyvin rajallinen, pitää kuitenkin kehittää jokin muu ratkaisu.

Kuten kuvassa 20 näkyy, mitta-arvo muodostaa kosinikäyrän osan mittauksen aikana. Kun lähestytään käyrän alaosassa sijaitsevaa todellista mittaa, alkaa mittatulosten välinen muutos pienentyä. Tätä tulosten välistä muutosta voidaan käyttää nopeuden korjaamiseen. Tässä työssä ei toteutettu adaptiivista nopeuden säätöä mittausliikkeelle, mutta sen toteuttaminen ei ole erityisen monimutkaista. Perusperiaatteena on, että mitä vähemmän mitta-arvo muuttuu, sitä hitaammaksi robotin liikenopeus lasketaan. Käytännössä pinnankarheuksien ja mittalaitteen kohinan vuoksi on todennäköisesti tarpeen keskiarvoistaa muutoksien määrää liukuvalla keskiarvolla. Koska liike tehdään robotin työohjelmassa eikä Karel-ohjelmassa, pitää nopeuden muuttaminen tehdä esimerkiksi järjestelmämuuttujan `$MCR[1].$prgoverride` avulla (Fanuc Robotics 2006, 8–22).

Toinen mahdollisuus tarkkuuden parantamiseen on tehdä mittausliikkeestä kaksivaiheinen. Ensin nopea pyyhkäisyliike, missä etsitään mitan minimikohta kar-

keasti. Sen jälkeen palataan minimikohdan alapuolelle, josta noustaan hieman minimikohdan yläpuolelle hitaasti. Tämän toteuttaminen on hieman edellistä haastavampaa ja on kyseenalaista, saavutetaanko tällä varsinaisesti mitään hyötyä.

Todennäköisesti paras tapa kuitenkin on valita mittausliike niin, että liikematka ei ole tarpeettoman pitkä. Näin voidaan valita alusta alkaen riittävän pieni liikenopeus ja saavutetaan riittävä tarkkuus ilman ohjelmamuutoksia.

9.3 TESTAUS JA TESTITULOKSET

Mittauksessa testattiin neljää asiaa:

- epäsymmetriset kappaleet
- kestävyys ja toistotarkkuus
- mittausnopeuden vaikutus eri pinnankarheuksilla
- ympäristövaikutukset mittauslaitteen toimintaan.

Menetelmä

Mittalaitteena oli Mitutoyo Bore Gages 511-sarjan reikäindikaattori Mitutoyon ABSOLUTE Digimatic Indicator ID-C 543-sarjan mittakellolla. Reikäindikaattori oli kiinnitetty telineeseen ja mitattava kappale tuotiin mittalaitteelle robotin avulla. Mittausliike tehtiin robotilla ja mittaustulos kirjattiin. Reikäindikaattorin tarkkuudeksi Mitutoyon nettisivuilla on ilmoitettu 5 µm ja kappaleiden materiaaliksi arvioitiin S355mc.

Ensimmäisessä kokeessa tulokset kirjattiin käsin paperille ja mittakellon nollattiin manuaalisesti jokaisen mittaliikkeen välillä. Yhdestä kappaleesta otettiin (pinnankarheus) yhteensä 15 mittaustulosta. Aluksi kappaleesta otettiin 7 mittaustulosta neljällä eri nopeudella (T1-20°/sec: 30%, 60%, 100% sekä T1 max (200mm/s)). Sitten kappaletta käännettiin 90° ja siitä otettiin vielä 8 mittausta. Kappaletta käännettiin 90°, jotta saataisiin selville sen mahdollinen soikeus. T1 (Teach-1) -tila rajoittaa robotin liikkeet 200mm/s (100 %) maksiminopeuteen.

Mittausten jälkeen vaihdettiin kappaletta ja toistettiin mittaukset samalla menetelmällä. Mittalaitteen mittapäätä jouduttiin vaihtamaan mitattavan kappaleen sisähalkaisijan mukaan. Kolmannen kappaleen mittaamiseen käytettiin kahta mittapäätä (5 mm pituusero) sekä 3 mm ja 2 mm tarkkuusprikkoja.

Pienin kappale mitattiin 40 mm (huom. ei mittapään pituus vaan mittauskategoria) mittapäällä ja 3 mm prikalla. Keskikokoinen 45 mm mittapäällä ja suurin 45 mm mittapäällä ja 2 mm prikalla.

Ensimmäisessä kokeessa tutkittiin ainoastaan mittaustoistotarkkuutta, joten kappaleiden tarkkoja absoluuttisia mittoja ei tiedetty.

Kappaleiden pinnakarheudet (3) mitattiin erillisellä Mitutoyo Surftest SJ-201 Series pinnankarheusmittalaitteella. Saadut Ra-arvot kirjattiin ylös.

Ennen mittaussarjaa otettiin ylös mitattavan tilan lämpötila sekä mittalaitteen lukema lepotilassa. Ensimmäisessä kokeessa todettiin myös, että mittalaitteen lukema lepotilassa ei aina palautunut samaan arvoon mittausten välissä. ”Nolla-arvo” heittelehti maksimissaan 2 µm.

Toisessa kokeessa tutkittiin kappaleiden sisähalkaisijoiden absoluuttista mittaa. Tätä varten mittalaite piti kalibroida käyttäen kalibrointirengasta (61.992 mm). Mittauksessa käytettiin vain hidasta nopeutta 5°/sec parhaimman tuloksen saamiseksi.

Nollapiste asetettiin tähän (61,992 mm) lukemaan, ja näin eliminoitiin virhe, joka aiheutui siitä, että mittalukema ei aina ollut sama ääriasennossa. Kalibrointirenkaalla tarkistettuna mittalukema vaihteli välillä 61,992 - 61,990, eli kalibroinnilla ei tullut aina samaa lukemaa. Virheen arvioitiin johtuvan mittakellon rakenteesta ja kiinnitystavasta, tarkkuudesta, kalibrointirenkaan sisäpinnan karheudesta ja lämpölaajenemisesta sekä siitä, että kalibrointirengasta käytettiin mittalaitteessa käsin.

Ensimmäisen kappaleen (Ra: 10,18) mittauksessa käytettiin samaa kalibrointia kuin kalibrointirenkaan kanssa (paras tarkkuus). Toista kappaletta (Ra: 8,27) mitattaessa jouduttiin poistamaan 2 mm prikka. Prikan paksuus mitattiin mikrometrillä 2,010 mm, tämä määrä vähennettiin myös mittaustuloksista. Muuten kalibrointi pidettiin samana. Kolmatta kappaletta (Ra: 2,25) ei pystytty mittaamaan samoilla tarkkuuksilla, sillä sen mittaamiseen olisi jouduttu vaihtamaan mittapäätä 5 mm lyhyemmäksi. Mittapään pituuseroa ei pystytty varmentamaan mikrometrin tarkkuudella (ulkomuodon vuoksi). Tälle mittausalueelle ei myöskään ollut kalibrointirengasta, jotta mittaus olisi voitu tehdä uudella kalibroinnilla.

Robotin ohjelmointi

Robottiohjelman tehtäväksi määriteltiin mitattavan kappaleen siirtäminen mittalaitteelle. Ohjelmoinnin ensimmäinen vaihe oli paikoittaa kappale tarkasti niin, että robotin leuat eivät osuisi mitattavan kappaleeseen hakuvaiheessa. Leuat aukesivat vain hieman enemmän kuin kappaleeseen tarttumiseen vaadittiin, joten paikoitus oli tärkeää. Paikoitus toteutettiin niin, että robotti työnsi kappaletta pienen matkan pitkin aihiopöytää työkalun kahdella leualla. Paikoituksen jälkeen robotti tarttui kappaleeseen, nosti sen ja siirsi kappaleen lähelle mittausasemaa samalla kääntäen sen niin, että kappale pääsi leuat avautuessa asettumaantyökalun pohjalle. Näin kappaleen asema työkalussa saatiin määriteltyä tarkasti. Seuraavaksi ohjelmassa tehtiin mittausliikkeet. Mitattava kappale nostettiin sopivassa kulmassa suoraan ylöspäin niin, että mittalaite osui kappaleen sisään. Sitten kappaletta työnnettiin hieman mittalaitetta kohti niin, että mittapää

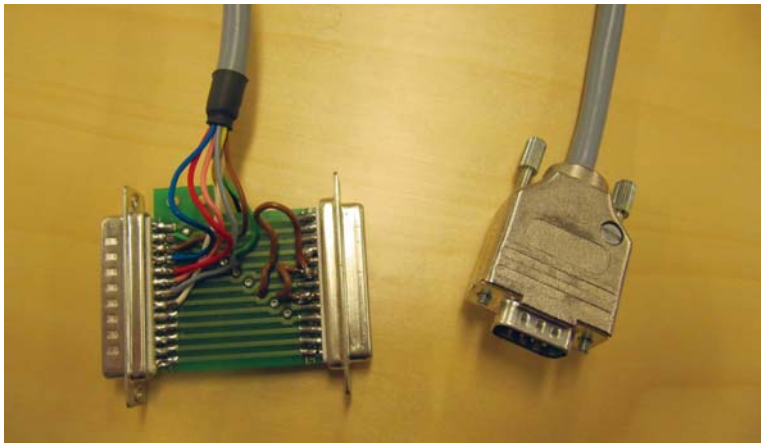
upposisyvemmälle kappaleeseen. Tämän jälkeen kappaletta käännettiin niin, että mitattavan kappaleen pituusakseli ohitti mittalaitteen pituusakselin. Mittalaite tallensi pienimmän arvon eli mitattavan kappaleen halkaisijan. Mittausliikkeessä nopeutta säädettiin ensimmäisessä kokeessa käsiohjaimella. Myöhemmin mittausliikkeen nopeus sidottiin rekisteriin ja vaihdettiin liikenopeuden yksiköksi astetta/sekunnissa. Ohjelmaan lisättiin silmukka, jotta robotti toistaisi vain mittausliikkeitä. Ohjelmassa mittausliikkeiden jälkeen robotti vie kappaleen takaisin aloitusasemaan samaa reittiä, jättää kappaleen ja siirtyy aloituspisteeseen.

Toisessa kokeessa ohjelmaa parannettiin niin, että robotti teki automaattisesti 10 toistoa. Lisäksi tiedonsiirron ollessa käytössä se kirjasi mittaustulokset automaattisesti tekstitiedostoon muistitikulle. Mittalaitetta ei tarvinnut nollata, sillä se asetettiin liukuvaa mittaustilaan, ja robotin ohjelma tallensi aina pienimmän mitatun arvon. Testien aikana oli käytössä T1-tila.

Ohjelmaa ei päästy testaamaan automaattitilassa, sillä robottisolu oli miehitetty testien aikana, eikä robottia voinut turvallisuussyistä käyttää maksiminopeudella.

Tiedonsiirto

Mittalaitteen ja robotinvälisen tiedonsiirron mahdollistamiseksi rakennettiin nollamodeemikaapeli (9-napainen). Aluksi etsittiin tietoa nollamodeemikaapelista ja RS232-kaapelista Internetistä. Löydettyjen tietojen ja kytkentäkaavioiden avulla rakennettiin testikaapelin robotin ja mittalaitteen välille (kuva 19).



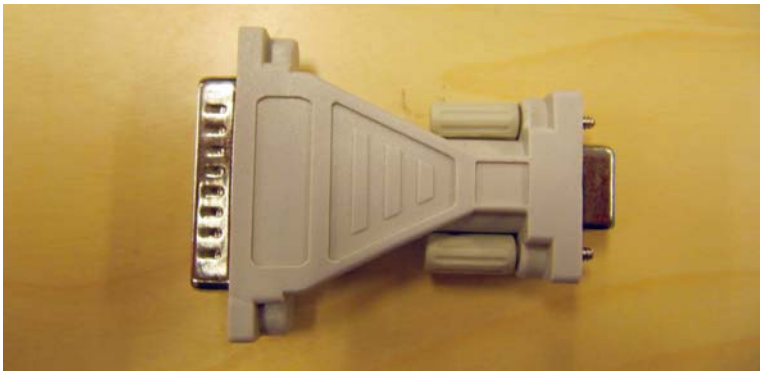
KUVA 19. *Robotin ja mittalaitteen välisen testikaapelin kytkentä.*

Testikaapeli alkoi toimia halutulla tavalla, kun rakennettuun kytkentään lisättiin vielä yksi siltaus.

Seuraavana tehtävänä oli rakentaa varsinaisen kaapelin rakennettiin annetusta nollamodeemikaapelista ja 9-25-adapterista (kuva 20).

KUVA 20. *Robotin ja mittalaitteen välinen kaapeli muistuttaa tavallista tietokonekaapelia.*

Kaapelirakennettiin kahdesta 9-pinnin liittimestä (uros ja naaras) ja 8-napaisesta kaapelista. Tämä oli mahdollista, kun kaksi pinniä silloitettiin liittimessä ja liitettiin samaan johtimeen. Kaapeli rakennettiin samalla tavalla kuin testikaapeli Kyseinen kaapeli ei kuitenkaan toiminut, kun siihen liitettiin 9/25 adapteri. Sähkömittauksella todettiin, että adapteri oli ristikytkennällä, joten rakentamamme kaapelin tuli olla suorakytkennällä. Rakennettu kaapeli jouduttiin muuttamaan suorakytkentäiseksi. Tämän jälkeen kaapeli toimi halutulla tavalla 9/25 adapterin kanssa. Kaapelin ja adapterin välissä käytettiin uros/uros muunninta.



KUVA 21. *9/25 adapteri.*

KUVA 22. *Uros-naaras -jatkopala.*

KUVA 23. *Kaapeli kytkettynä robottiin.*

Kaapeli asennettiin kiinteästi robottisoluun. Kaapeli mahdollisti tiedonsiirron mittalaitteen ja robotin välillä. Kaapelin ansiosta mittaustulokset pystyttiin kirjaamaan sähköiseen muotoon.

Tulos

TAULUKKO 2. *Suurimmat poikkeamat toistotarkkuudesta.*

Poikkeamat	**2,25**	**8,27**	**10,18**
30 %	0,000	0,010	0,010
60 %	0,001	0,018	0,022
100 %	0,002	0,037	0,026
T1 max	0,005	0,030	0,041

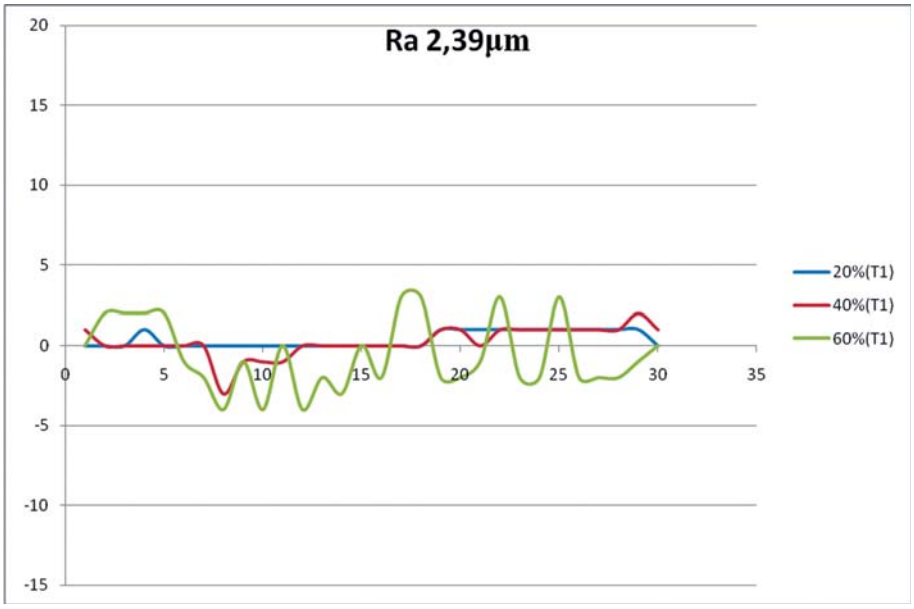
Huomiot ja päätelmät

Todettiin, että lämpötila ei vaikuta toistotarkkuuteen, kun kaikki mittaukset tapahtuvat samassa tilassa ja lämpötilassa (kun lämpötilat ovat tasoittuneet). Mittalaitteen mittapää saattaa naarmuttaa mitattavan kappaleen sisäpintaan uran, jos materiaali on pehmeää ja toistoja on useita samassa asennossa.

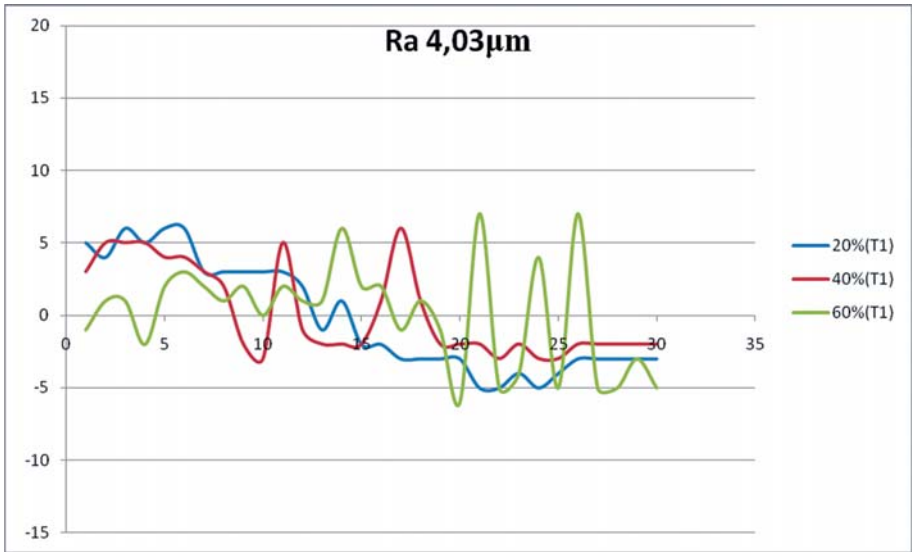
Testit aloitettiin pinnankarheuksien testaamisella. Käytössä oli kolme erilaista kappaletta, jotka sorvattiin tiettyihin pinnankarheuksiin. Näillä kappaleilla ajettiin kolmella eri nopeudella 30 toiston mittaustestit. Hajontaa voitiin vertailla nopeuden kasvaessa.

Testitulokset

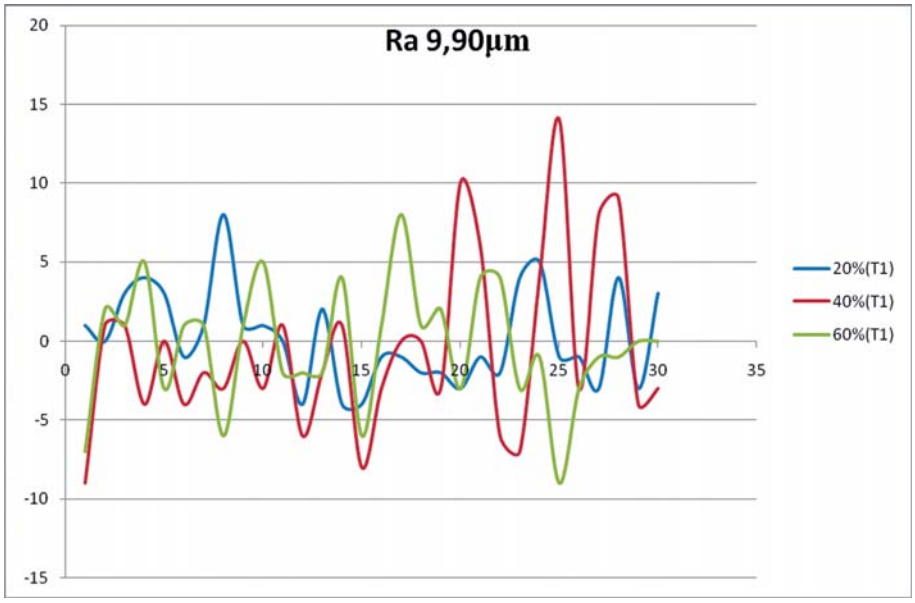
Ryhmä testasi mittaustulosten hajontaa kolmella eri pinnankarheudella mittausliikkeen nopeuden muuttuessa. Kaikki pinnankarheudet kolmella eri nopeudella testattiin. Kuvaajissa nollakohtana toimii 30 mittausarvon mediaani. Pinnankarheudet olivat Ra 2,39 µm, Ra 4,03 µm, Ra 9,90 µm. Kuvioissa 6, 7 ja 8 on mittaustulokset esitetty kuvaajina.



KUVIO 6. *Pinnankarheus Ra 2,39 µm.*



KUVIO 7. *Pinnankarheus Ra 4,03 µm.*



KUVIO 8. *Pinnankarkeus Ra 9,90 µm.*

Noin 10 toiston jälkeen mittalaite oli kuluttanut kappaleisiin niin suuren uran, että mitta-arvot muuttuivat pienemmiksi. Muutoksen huomaa hyvin esimerkiksi kuviossa 8 (Ra 4,03µm) 20 % viivalla. Tästä syystä mitattavia kappaleita käännettiin muutama aste jokaisen 30 sarjan jälkeen. Kuten voidaan huomata, mittausnopeuden kasvu suurentaa hajontaa varsinkin pienillä pinnankarheuksilla. Toisaalta heitot ovat vielä Ra 9,90 µm:kin karheudella vielä hyvin pieniä (±0,0014 mm).

KUVA 24. *Mittalaite sivulta kiinnitettynä.*

KUVA 25. *Nivelosat ja kiinnitys.*

KUVA 26. *Mittauspää.*

KUVA 27. *Mittalaitteisto kokonaisuudessaan.*

10 OPTINEN MIKROMETRI PYÖRÄHDYSKAPPALEIDEN ULKOHALKAISIJOILLE

Lähteinä Sakari Koivusen opinnäytetyö sekä Antti Meriön ja Gaius Voltin projektiraportti.

Tavoitteena oli kehittää koneistettujen kappaleiden robotisoitu ulkohalkaisijan tarkastusmittausjärjestelmä. Menetelmä sisältää myös takaisinkytkennän eli työkalun korjausparametrien automaattisen asettamisen kappaleet koneistavaan työstökeskukseen.

10.1 LÄHTÖTILANNE

Koneistuksen alihankintayritys haluaa automatisoida sarjatuotantona valmistettavien koneistettujen kappaleiden laadunvalvontaa.

Kappale valmistetaan robotisoidussa koneistussolussa. Tällä hetkellä laadunvalvonta suoritetaan käsin, ja koneistuskeskuksen asetuksia korjataan tarpeen mukaan käsin.

Mitattavat kappaleet ovat pyörähdyssymmetrisiä, sorvaamalla koneistettuja kappaleita. Kappaleet ovat 20–180 mm pitkiä ja niiden halkaisijat ovat leveydeltään 60 350 mm. Robotin tartunta useimpiin kappaleisiin tapahtuu ulkopuolelta, osaan kuitenkin sisäpuolelta.

Kappaleiden halkaisijatoleranssi on ±0,01 mm. Toleranssiaseman pohjalta korjataan koneistuskeskuksen työkaluparametreja. Jos mitattava kappale ei täytä toleranssivaatimusta, se hylätään ja poistetaan robotilla materiaalivirrasta.

Mittaukset suoritetaan normaaleissa konepajaolosuhteissa, joten mittausmenetelmän häiriöalttiuteen ja olosuhteiden muutosten sietoon pitää kiinnittää normaalia enemmän huomiota. Myös miehittämättömän koneistuksen vaatimukset on huomioitava.

Alkuvaiheessa tutkittiin erilaisia mittaustapavaihtoehtoja. Optisten mikrometrien lisäksi vaihtoehtoina tutkittiin muita kosketuksettomia mittaustapoja kuten pinnasta mittaavia säteen heijastumiseen perustuvia lasermittalaitteita, erilaisia lähestymisantureita ja myös kosketukseen perustuvia antureita.

Tutkimuksessa päädyttiin valitsemaan optinen mikrometri suuren mittaustarkkuuden ja kappaleen vapaamman sijoittelun perusteella. Optinen mikrometri ei myöskään aseta rajoituksia esimerkiksi mitattavan uran leveydelle, materiaalille tai pinnanlaadulle.

10.2 TUTKITUT MITTALAITTEET

Projektiin liittyen testattiin erilaisia optisia mikrometrejä sopivan mittauskomponentin löytämiseksi. Aluksi testattiin toimittajalta käyttöön saatua Omron ZX-GT-lasermikrometria Smart-Monitor GT-ohjelmistoineen. Toinen tutkittava mittalaite oli toimittajalta käyttöön saatu Keyence LS-7030M-led-mikrometri ja näyttölaite LS-7501 sekä LS Navigator-tietokoneohjelmisto. Samantyyppisiä laitteita löytyy myös muutaman muun valmistajan valikoimista. Omronin ja Keyencen lisäksi Mitutoyolla on maahantuoja Suomessa. Kaikilla valmistajilla ei kuitenkaan ole maahantuojaa Suomessa.

Alustavien mittalaitetestausten ja saatavuuden selvittämisen jälkeen päädyttiin valitsemaan projektin seuraaviin vaiheisiin Keyence-mikrometri. Korkeamman hankintahinnan vastapainona ovat merkittävästi parempi tarkkuus, suurempi mitattavan kappaleen maksimihalkaisija ja säätötarpeen kuten esimerkiksi suuntauksen vähäisyys.

10.2.1 Mittaustapa

Kappaleen mittausta tutkittiin suoraan robotin pitelemänä tai rakennettavan mittauskoneen kannattelemana. Mittaus robotin pitelemänä vaatisi mittalaitteelta suuremman tarkkuuden ja etenkin mittausnopeuden. Tämä johtuu siitä, että robotin ja tartunnan epätarkkuuden vuoksi mittaus on suoritettava kappaleen liikkuessa. Toisaalta ratkaisu vaatisi paljon vähemmän tilaa koneistussolussa. Myös materiaali- ja rakentamiskustannukset olisivat merkittävästi alhaisemmat. Erillisen mittauskoneen etuja olivat mittaustarkkuuden helpompi hallinta, mahdollisuus sulkea mittaustapahtumaan vaikuttavat ulkoiset tekijät mittauskoneen ulkopuolelle ja mahdollisuus, että robotti on vapaa tekemään muita tehtäviä mittauksen aikana.

Alkuvaiheen mittauslaiteanalyysien ja -testien perusteella päädyttiin jatkamaan kehitystyötä erillisen mittauskone-vaihtoehdon pohjalta. Vaatimusten pohjalta suunniteltiin mittauskone.

Projektin edetessä saatiin kuitenkin hyvin rohkaisevia testaustuloksia Keyence-mittalaitteesta. Samalla havaittiin robotin ja vaaputuksen edullisuus kosinivirheen hallinnassa. Tämän jälkeen päätettiin jatkaa projektia robottia hyödyntävänvaihtoehdon pohjalta.

10.2.2 Kappaleen asento mitattaessa erillisen mittauskoneen avulla

Seuraavaksi tutkittiin erilaisia kappaleen mittausasentoja. Vaaka-asento vaatisi joko pitkittäisen puristavan otteen tai kannattimet. Pystyasento vaatii vähintään yksinkertaiset kannattimet, toki myös puristava ote olisi mahdollinen. Puristava ote vaikeuttaisi asennosta huolimatta selvästi kappaleen tuomista robotilla mittalaitteelle, sillä robotin tarttujien päätyote hankaloittaa pitkittäisen puristavan otteen muodostamista mittauskoneessa. Puristavan otteen aikaansaamista vaikeuttavat myös sisä- tai keskiöreiän puuttuminen sekä sisäreiän erilaiset halkaisijat eri kappalemallien välillä.

Vaaka-asento kannattimien päällä aiheuttaisi ongelmia monimutkaisten kappaleiden kanssa. Samalla voisi muodostua tilanteita, joissa tukien sijoittelu estäisi halutun mittauslinjan käytön. Suuren halkaisija-pituussuhteen yhdistelmä on ongelmallinen vaaka-asennossa. Vaaka-asennon etuna voi katsoa olevan parempi soveltuvuus pitkille, ohuille akselimaisille kappaleille. Puristava vaakaote ja pystyasento puristuksella tai ilman mahdollistavat kaikki myös kappaleen pyörityksen lisäämisen myöhemmin. Pyöritys tarvitaan, jos halutaan myöhemmin laajentaa mittausta myös kappaleen pyöreyden analysointiin.

Päädyttiin valitsemaan pystyasento, sillä mitattavat kappaleet ovat pääasiassa holkkimaisia kappaleita, joiden pituus-halkaisijasuhde on melko pieni.

10.2.3 Kappaleen mittaus robotin pitelemänä

Koska robotin paikoitustarkkuus on heikompi kuin mittauskoneella, etenkin kulman suhteen, mittaus robotin paikallaan pitämästä kappaleesta ei olisi järkevä vaihtoehto. Otteen vaihtelun ja robotin kulmapaikoitustarkkuuden aiheuttamien vaihteluiden seurauksena syntyisi mittaustarkkuuden tasolla haitallisen suuri kosinivirheen mahdollisuus.

Kosinivirheen minimoimiseksi mittaus on syytä suorittaa niin, että mitattavaa kappaletta vaaputetaan mittalaitteen anturivälissä. Samalla mittalaite mittaa halkaisijaa. Asetuksena minimimitan pito (Automatic Bottom Hold). Näin mittalaite pystyy mittaamaan mahdollisimman tarkan halkaisijamitan, ja kosinivirheen vaikutus saadaan samalla minimoitua jopa paremmin kuin erillistä mittauskonetta käyttämällä olisi ilman erikoisratkaisuja kyetty.

Robotin vaaputusliikettä ohjelmoitaessa on kiinnitettävä erityistä tarkkuutta törmäysvaaraan suurten halkaisijavaihteluiden läheisyydessä mitattaessa. Vaikka kappale muutoin mahtuisikin mittalaitteen anturiväliin, mittalaitteen runkopalkin ohitse, liian laaja vaaputusliike voi aiheuttaa törmäyksen.

10.2.4 Vaaputusnopeus

Keyencen-mittauslaitteen toimintaa tutkittaessa havaittiin vaaputusnopeuden vaikutuksen mittaustulokseen olevan melko pieni. Sen sijaan mittauslaitteen keskiarvoistusasetuksen vaikutus oli erittäin merkittävä. Vaikka mittauslaitteen mittaustaajuus on yli 2000 Hz, esimerkiksi 1024 mittauksen keskiarvoitus aiheuttaisi jo noin viiden asteen (ts. noin ±2,5°) kulmamuutoksen yhden mittaustuloksen keruuaikana, jos vaaputusnopeus olisi 10°/s. Seurauksena olisi ulkoisen tarkkuuden selvä heikkeneminen.

Toinen huomioitava seikka on mitattavan kappaleen halkaisijan, tarkkuuden ja vaaputusnopeuden suhde. Pienemmällä halkaisijalla mittaustapahtuman aikaisen kulmamuutoksen kosinivirheen mitallinen muutos on pienempi, mutta suhteellisen tarkkuuden kannalta muutos voi olla jopa merkittävämpi kuin suurihalkaisijaisella kappaleella.

Mittauslaitteen testaushavaintojen pohjalta alle 100 mm halkaisijoilla keskiarvoistus on syytä pitää joka tapauksessa alle 128 asetuksen ja kulmanopeudet alle 5°/s. Suuremmilla halkaisijoilla ensisijaisesti keskiarvoistusta pitää pienentää. Toisaalta mittaustapahtuman kokonaiskestoajassa vaaputusliikkeen käyttämä aika on niin lyhyt, ettei vaaputusnopeutta ole sen vuoksi tarpeen nopeuttaa. Mittauslaitteen keskiarvoistusasetusta ei käytännössä ole tarpeen nostaa sisäisen tarkkuuden parantamiseksi. Siten keskiarvoistus voidaan asettaa esimerkiksi arvoon 32 ja robotin vaaputusliikkeen nopeudeksi ohjelmoidaan 1°/s–3°/s. Suuremmilla halkaisijoilla voidaan käyttää alhaisempaakin keskiarvoistusasetusta. Mittaustulosten toistettavuuden kannalta keskiarvoistusasetuksen alarajana voidaan kuitenkin pitää asetusta 4.

Asetusten kohdalta toistettakoon vielä, että vaaputettaessa pitää ehdottomasti käyttää minimimitan pitoa (Automatic Bottom Hold).

10.2.5 Kappaleen asento mitattaessa robotin avulla

Kun robotti pitelee kappaletta, niin kappaleen (pysty-/vaaka-) asento mitattaessa ei ole yhtä perustavaa laatua oleva kysymys. Mittausasento voidaan valita niin, että mittalaitteen anturit on helppo suojata laskeutuvilta epäpuhtauksilta, näkyvyys on hyvä robottia ohjelmoitaessa ja laitteisto on helppo rakentaa. Robotin pitelemänä mitattavan kappaleen ulkomitat eivät myöskään aseta samankaltaisia rajoituksia kuin mittauskone asettaisi. Halkaisijarajoitus riippuu tässäkin vaihtoehdossa pääasiassa käytettävästä mittalaitteesta tai mittalaitteista.

Robotin pitelemänä kappaleesta on melko helppo mitata samasta kappaleesta useamman mittauspisteen halkaisija.

Robottivaihtoehdossa robotin tartuntaleukojen ulkopuolinen ote voi vaikeuttaa mittauksia lähellä kappaleen otteenpuoleista päätä. Suurihalkaisijaisilla kappaleilla tartunta voi silti olla mahdollista mittausta estämättä. Ongelma voidaan välttää paitsi mittauskohtaa muuttamalla, myös tartuntaleukojen tai niiden sijoittelun uudelleensuunnittelulla.

Näkyvyys ohjelmoitaessa ja suojaus ilman epäpuhtauksilta ovat osin vastakkaisia vaatimuksia. Näkyvyys voidaan aikaansaada läpinäkyvällä koteloinnilla tai ohjelmoitaessa avattavilla kotelonosilla.

10.2.6 Mittauksen ohjaus

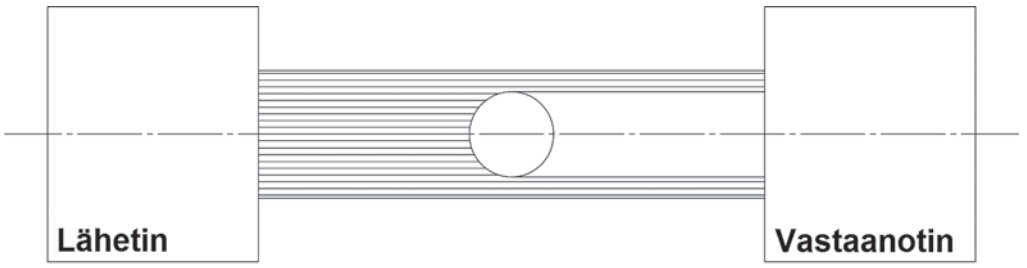
Mittaustapahtuman ohjaus ja mittaustietojen keräys voidaan kokonaisuudessaan suorittaa robotin ohjaimella. Mikäli tiedonkeruutarve on suurempi tai mittaustuloksille pitää suorittaa laajempaa analysointia, voi olla tarpeen viedä mittaustulokset automaattisesti tähän tarkoitukseen varatulle tietokoneelle. Tällöinkin on kuitenkin järkevää ohjata mittaustapahtuman kulkua robotin ohjaimella.

Mittauslaitteen keskeiset parametrit on syytä asettaa sähköisesti mittaustapahtumaa ohjaavalta laitteelta. Näin voidaan välttää asetusten odottamattomien muutosten aiheuttamat virheet. Samoin mitattaessa samasta kappaleesta useampia, halkaisijaltaan eri suuruisia kohtia, voidaan huolehtia siitä, että jokaisen mittauskohdan mukaiset asetukset ovat oikeaan aikaan asetettuna. Kun suurempia halkaisijoita mitattaessa mittauslaitteen anturiväliä muutetaan, voidaan huolehtia siitä, että asetukset ovat aina tilanteen mukaiset.

11 YLEISTÄ OPTISISTA MIKROMETREISTÄ

Lähteenä Antti Meriön ja Gaius Voltin projektiraportti.

Optiset mikrometrit perustuvat lähetinosan ja vastaanotinosan väliseen tarkasti yhdensuuntaiseksi suunnattuun ohueen valoverhoon eli telesentriseen toimintatapaan. Mitattava kappale tuodaan mittalaitteen mittausväliin. Mikrometri mittaa mittausvälissä kappaleen peittämää säteen osaa valoverhoon pituusakseliin nähden suorakulmaisessa tasossa. Mittalaitteen elektroniikka käsittelee ja muuttaa mittaustuloksen mittalaitteen asetusten mukaisesti numeroarvoksi.



KUVA 28. *Optisen mikrometrin toimintaperiaate (telesentrinen).*

Yleensä sama mittalaite pystyy mittaamaan myös valoverhon peitetyn tai peittämättömän osan mittaa. Tämä mahdollistaa suuren määrän jatkosovelluksia, mukaan lukien kahden anturin samanaikaisen käyttämisen suurempien halkaisijoiden mittaamiseksi.

Optisista mikrometreistä voidaan erottaa erilaisia toimintaperiaatteita:

- Pyyhkäisevissä lasermikrometreissä tuotetaan lähettimessä pyörivän monitahoisen peilin ja kollimaattorilinssin avulla yhdensuuntainen ja pyyhkäisevä lasersäde. Vastaanotinosassa säde muutetaan sähköiseksi signaaliksi. Halkaisija lasketaan vastaanottimen kennolle pääsevän ja kappaleen peittämän signaalin aikasuhteen avulla.
- Tasovalotusperiaatteella toimivissa (optisissa) mikrometreissä lähettimestä lähtevä, suodatettu ja yhdensuuntainen valosäde vastaanotetaan viivamaiselle (yksiulotteiselle) kamerakennolle.Kamerakennon tuottaman viivamaisen kuvan pikselien sijainnin ja intensiteetin avulla lasketaan halkaisija.

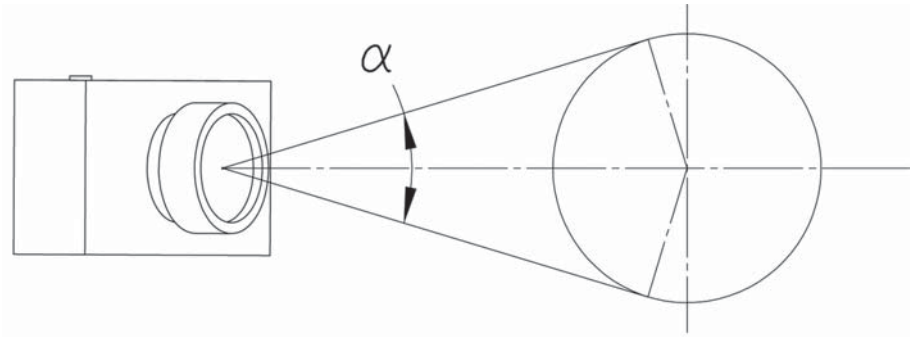
- Valon voimakkuuteen perustuvissa (optisissa) mikrometreissä lähettimistä lähtevä, suodatettu valonsäde vastaanotetaan kokoojalinssin kautta valoherkälle komponentille. Komponentin tuottaman signaalin jännitteen muutoksen avulla lasketaan mitattavan kappaleen halkai-sija.

Toimintaperiaate vaikuttaa tarkkuuden lisäksi laitteiden odotettavissa olevaan kestoikään. Pyyhkäisevien lasermikrometrien pyörivä peili laakerointeineen ja sen moottori ovat kuluvia osia. Muilla toimintaperiaatteilla toimivissa mikrometreissä ei tarvita vastaavia liikkuvia osia. Koska kamerakennoperiaatteella toimivien optisten mikrometrien mittaus perustuu paitsi pikselin binääriseen tilaan (valaistu/pimeä) myös reunapikselien intensiteettien analysointiin, mittauksen erottelukyky voi olla tarkempi kuin käytetyn kamerakennon resoluutiosta voisi yksioikoisesti olettaa.

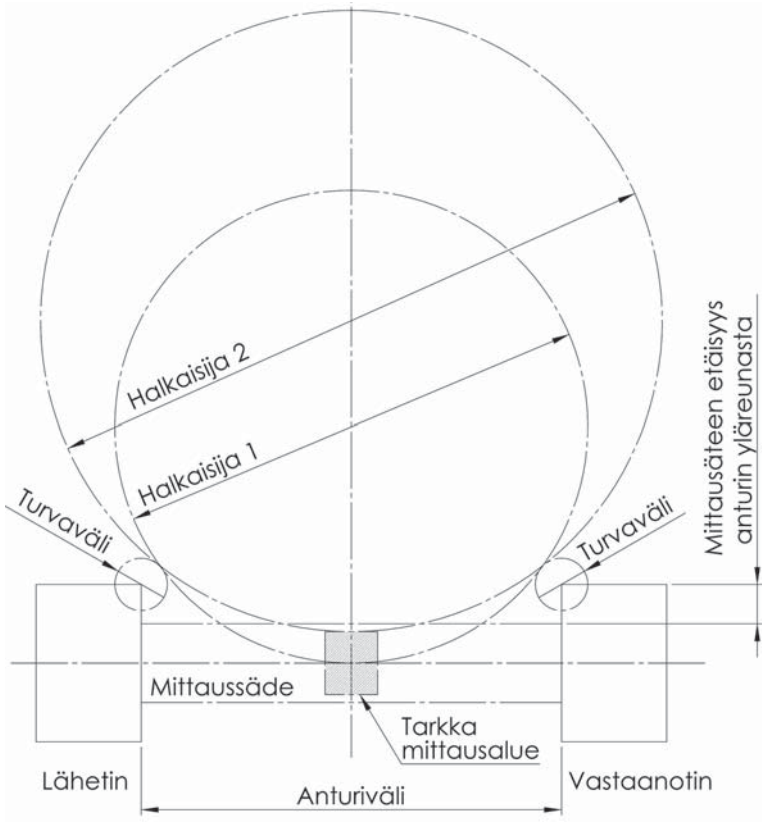
Yleensä samat laitteet pystyvät laskemaan myös muita mittoja kuin halkaisijoita, esimerkiksi seuraamaan kappaleen reunaa tai useampien mittausväliin ulottuvien tankojen halkaisijoita sekä reunojen sijainteja. Kaksi tai jopa useampia antureita voidaan myös yhdistää, jolloin voidaan esimerkiksi mitata suurempia halkaisijoita (vastakkaiset anturit) tai tarkastaa tangon pyöreys kahdesta suunnasta tehtyjen mittausten tuloksena (ristikkäiset anturit).

11.1 TELESENTRINEN LINSSI VAI KAMERA?

Optiset mikrometrit on syytä erottaa ei-telesentrisen linssin ja kameran pohjalta toimivista mittalaitteista. Optisen mikrometrin mittaus perustuu samansuuntaisiin valonsäteisiin. Kameran linssissä mittaussäteet puolestaan kulkevat saman optisen polttopisteen kautta, eivätkä valonsäteet ole yhdensuuntaisia. Käytännössä tästä seuraa perspektiivivirhe kolmiulotteisia kappaleita kuvattaessa. Perspektiivivirheen vaikutusta voidaan minimoida, mutta sen vaikutuksen poistaminen kokonaan mittaustilanteesta vaatisi joko epärealistisen pitkän polttovälin tai monimutkaisen kameran liikuttelulaitteiston. Molemmat ratkaisut toisivat toisaalta mukanaan uusia epävarmuustekijöitä.



KUVA 29. *Kamera ja tankomainen kappale.*



KUVA 30. *Mitattavan kappaleen maksimihalkaisija.*

11.2 MITATTAVISSA OLEVAT MAKSIMIHALKAISIJAT

Mitattaessa halkaisijaa yhden optisen mikrometrin avulla mitattavan kappaleen maksimihalkaisija on mikrometrin valoverhon tarkan mittausalueen leveys vähennettynä kappaleen sijoitustarkkuudella. Mikrometrien anturiväli on aina tätä suurempi, joten se ei muodosta rajoituksia yhden anturin tapauksissa.

Kahdella optisella mikrometrillä mitattaessa tilanne muuttuu. Vaikka optisen mikrometrin kiinteästä anturivälistä (esimerkiksi 160 mm) voisi helposti päätellä, että maksimaalinen mitattava halkaisija olisi sama, todellisuus on kuitenkin hieman toinen. Koska mikrometri mittaa etäisyyttä valoverhon reunasta, voi ajatella valoverhon ja mitattavan kappaleen kaarevan pinnan leikkaavaa aluetta ympyrän segmenttinä. Tällöin samalla anturivälillä voidaankin mitata merkittävästi halkaisijaltaan suurempia kappaleita. Maksimihalkaisijaa (Kuva 35: Halkaisija 1 tai Halkaisija 2) määritettäessä pitää huomioida paitsi anturiväli myös mikrometrien tarkan mittausalueen etäisyys anturien yläreunasta. Huomiota on kiinnitettävä myös kappaleen paikoitustarkkuuteen ja turvaetäisyyteen.

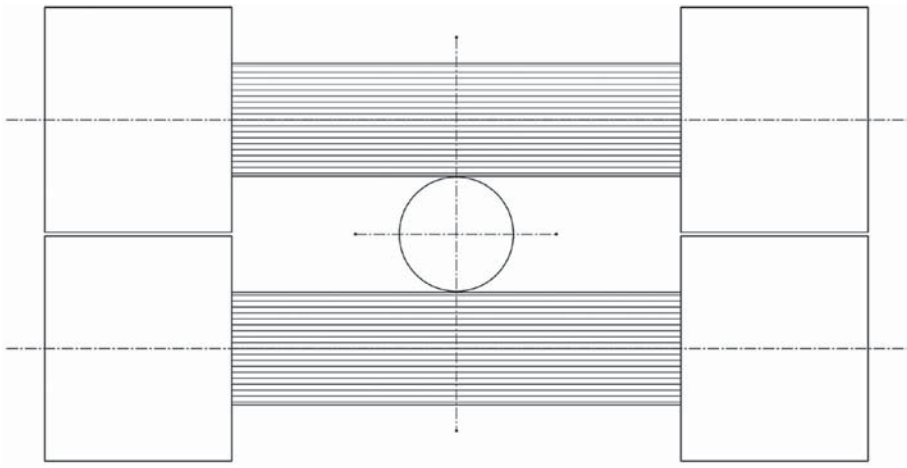
Joillakin anturimalleilla anturiväli ei ole kiinteä. Tällöin ei ole samanlaisia rakenteellisia rajoituksia maksimihalkaisijassa. Pitää kuitenkin huomioida tarkkuuden heikkeneminen saman anturin lähettimen ja vastaanottimen välisen etäisyyden kasvaessa. Joillakin mittalaitteilla anturivälin piteneminen hankaloittaa myös merkittävästi anturien kohdistamista.

11.3 MITATTAVISSA OLEVAT MINIMIHALKAISIJAT

Yksittäisen anturin mitattavissa oleva minimihalkaisija riippuu käytettävästä anturista. Tieto löytyy mittalaitteen valmistajan anturikohtaisista tiedoista.

Käytettäessä mittaamiseen kahta anturia tilanne ei ole sama kuin yhtä anturia käytettäessä. Mikrometrien tarkan mittausalueen etäisyys anturien koteloiden yläreunasta asettaa rajoituksen. Kahdella anturilla mitattaessa antureiden valoverhojen tarkkojen mittausalueiden väliin jää 'musta aukko'. Joillakin anturityypeillä tämä voi aiheuttaa tilanteen, jossa yhdellä anturilla mitattavissa olevan maksimihalkaisijan ja kahdella anturilla mitattavissa olevan minimihalkaisijan väliin jää halkaisija-alue, jota ei kyetä mittaamaan.

'Mustaa aukkoa' olisi teoriassa mahdollisuus pienentää sijoittamalla anturit pitkittäissuunnassa anturien koteloinnin verran eri kohtiin, ja samalla melkein puolittamalla katvealueen syvyys. Käytännössä tämä ei kuitenkaan ole mahdollista, sillä silloin mittalaitteen antureiden suurimman tarkkuuden alueet eivät olisi kohdakkain. Seuraus olisi mittaustarkkuuden heikkeneminen, kun toinen tai molemmat anturit mittaisivat tarkan mittausalueen ulkopuolella. Samalla myös kahdella anturilla mitattavissa oleva maksimihalkaisija pienenisi merkittävästi.



KUVA 31. *Mitattavissa olevan halkaisijan alaraja kahdella anturilla mitattaessa.*

11.4 MUITA HUOMIOITA

Keyencen kaikkien anturien mittauslinja ei ole sivuttaissuunnassa tarkasteltaessa anturien keskilinjalla, vaan muutaman millin sivussa. Tämä pitää huomioida anturien kiinnityksessä etenkin käytettäessä mittaukseen kahta anturia. Yksi vaihtoehto ongelman välttämiseksi on asentaa anturit niin, että toisen anturin lähetinosa ja toisen vastaanotinosa ovat samassa päässä ja päinvastoin. Tällöin anturien epäkeskeisyys jää samalle puolelle.

Keyencen joidenkin mikrometrimallien videokameraominaisuus ja mittauskohdan näyttö näyttölaitteen kuvaruudulla helpottavat oikean mittauskohdan löytymistä ja anturilinjan kohdistamista käytettäessä mittaukseen kahta anturia.

11.5 RAKENNE YHTÄ ANTURIA KÄYTETTÄESSÄ

Kun yhden anturin suurin mitattava halkaisija riittää kaikille mitattaville kappaleille, mittausanturi kiinnitetään tukevasti koneistussoluun. Anturi suojataan mittauskappaleen tuontisuunnasta katsottuna esimerkiksi tukevalla levyllä, joka estää mitattavan kappaleen tai robotin osien osumisen mittausanturiin virheellisen paikoituksen tapahtuessa. Suojalevy voidaan myös varustaa törmäyksen tunnistavalla turvakytkimellä, joka pysäyttää robotin ohjelman suorituksen törmäystilanteessa. Suojalevy ei kuitenkaan saa rajoittaa tai estää robotin vaaputusliikettä mitattaessa.

Mittausanturi suojataan yläpuolelta katoksella, joka estää laskeutuvien epäpuhtauksien pääsyn mittausanturille.

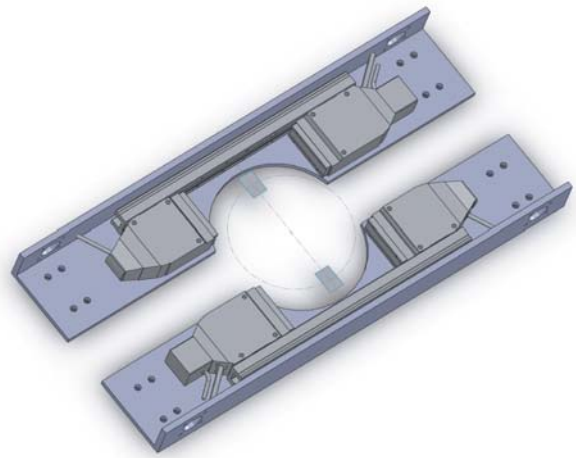
11.6 RAKENNE KAHTA ANTURIA KÄYTETTÄESSÄ

Haluttaessa mitata suurempihalkaisijaisia kappaleita on tarpeen käyttää kahta mittausanturia. Tästä huolimatta myös pienempiä halkaisijoita voidaan edelleen mitata yhdellä anturilla. Anturit voidaan kiinnittää tukeviin ja antureita suojaaviin L-profiiliteräksiin, kappaleen tuontisuunnasta katsottuna profiilin taakse. Terästen keskiosaan, mittausanturien tarkimman mittauskohdan kohdalle, koneistetaan kolo, jonka säde on hieman suurempi kuin suurimman mitattavan kappaleen säde. Kolo ulottuu niin syvälle, ettei se peitä tarvittavaa osaa mittausvaloverhosta. Tästä huolimatta mittausanturi ei saa näkyä mitattavan kappaleen tuontisuunnasta katsoen. Näin anturit voidaan suojata useimmilta ohjelmointivirheiltä ja virheellisen tarttumaotteen aiheuttamilta törmäyksiltä.

Mittausanturien teline on syytä tehdä sen verran irtonaiseksi, että törmäystilanteessa koko teline väistää, sen sijaan että aiheutuisi muodonmuutoksia. Tästä huolimatta telineen sijainti pitää pystyä palauttamaan tarkasti samaksi, jotta

robottiohjelman kohdistuksia ei tarvitse etsiä uudelleen. Telineeseen on syytä asentaa turvakytkin, joka pysäyttää robottiohjelman suorituksen mittalaitteen siirtyessä pois paikoiltaan (=törmäystilanne). Kuvassa 32 on kuvattuna anturien kiinnitys profiileja hyväksi käyttäen.

Mittausanturit on syytä asentaa eri suuntiin mittaaviksi. Tällöin ensimmäisen anturin lähetinosa ja toisen anturin vastaanotinosa ovat samassa päässä. Vastaavasti anturin 2 lähetinosa ja anturin 1 vastaanotinosa ovat samassa päässä. Näin vältetään joidenkin anturien mittauslinjan epäkeskeisyyden vaikutus. Anturien tarkin mittausalue pitää kuitenkin kohdistaa samalle linjalle.



KUVA 32. *Antureiden kiinnitys profiileihin.*

11.7 LINJAUKSET

Molempien mittausanturien mittaussäteiden taso pitää kohdistaa mahdollisimman tarkasti samalle tasolle. Perustilanteessa anturit kiinnitetään edellä mainittujen profiilien tarkasti tasomaisiksi koneistettuihin pintoihin niin, että anturien sivut ovat tasoa vasten. Anturien kiinnitysprofiilien yhdensuuntaisuuden pitää säilyä antureiden etäisyyssäädöstä huolimatta.

Anturien mittaustason yhdensuuntaisuuden tarkastaminen on mahdollista myös joidenkin anturimallien sisältämällä videokameralla. Suorapäätyisen kappaleen lähestyessä hyvin hitaasti suoraan mittausväliä näytöltä voidaan nähdä miten suuri ero on eri kohtien tulohetkessä mittausväliin. Tarkastus on syytä suorittaa paitsi mittausvälin keskellä myös mittausvälin molemmissa päissä.

11.8 ETÄISYYDEN SÄÄTÖ

Jos kaikki halutut suuremmat kappaleet voidaan mitata yhdellä anturivälillä, L-profiiliterästen välinen etäisyyden säätö voidaan toteuttaa yksinkertaisesti kahdella pitkällä säätöruuvilla. Asetus suoritetaan kerran ja lukitaan paikoilleen.

Jos anturivälin säätö on tarpeen toteuttaa ohjelmallisesti säädettävänä, se voidaan tehdä kahdella kuulamutterijohteella. Johteiden käyttö on helpointa toteuttaa yhdellä synkronimoottorilla, jolloin johteiden välisen välityksen pitää olla mahdollisimman välyksetön ja joustamaton. Niin säätöruuvien, kuulamutterijohteiden kuin välitystenkin välysten vaikutukset minimoidaan niin, että haluttua etäisyysasetetusta lähestytään aina samasta suunnasta.

Kun anturiväli on säädettävä, toinen anturi on kuitenkin syytä pitää liikkumattomana koneistussoluun verrattuna. Näin yhdellä anturilla mitattavat kappaleet voidaan aina tuoda samaan paikkaan.

11.9 YHDENSUUNTAISUUDEN TARKISTAMINEN JA MITTAUSLAITTEEN KALIBROINTI

Anturien mittaussäteiden samansuuntaisuuteen mittaussuunnassa pitää kiinnittää erityistä huomiota. Yhdensuuntaisuus voidaan tarkastaa mittaamalla saman kappaleen halkaisijaa mittausvälin molemmissa päissä. Jos mittaustulokset poikkeavat toisistaan, niin anturiväli ei ole tasainen. Tarkastusmittauksessa pitää kiinnittää erityistä huomiota kosinivirheen välttämiseen. Vaikka mittaustarkkuus anturivälin ääripäissä ei ole aivan paras mahdollinen, tarkastustapa on kuitenkin riittävä. Mittalaitteen tarkan mittausalueen kohdalla säädön vaikutus on paljon parempi.

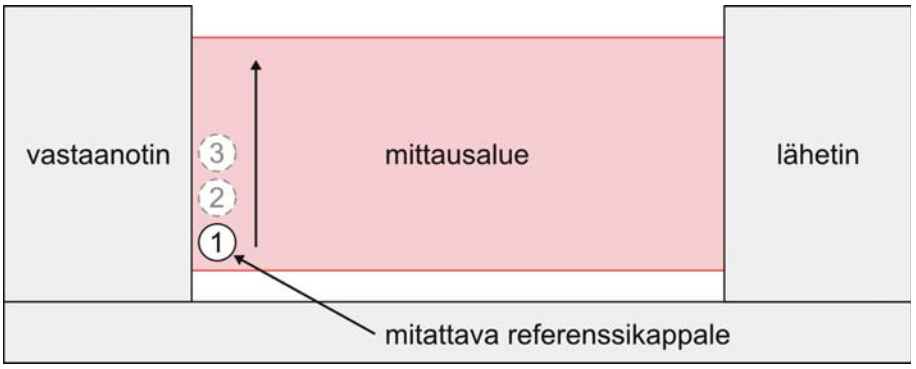
Kun yhdensuuntaisuus on säädetty ja tarkastettu, mittauslaite kalibroidaan niin, että antureiden väli tulee oikein huomioiduksi. Tällöin kahden mittausanturin käytöstä huolimatta mittauslaitteiden näyttö ilmoittaa yhden mittaustukoksen, joka edelleen hyödynnetään.

12 OMRON ZX-GT

Kappaleiden mittaamiseen testattiin myös Omron ZX-GT -mittalaitetta. ZX-GT on lasermikrometri, jonka mittausalueen leveys on 28 mm ja anturien välinen etäisyys maksimissaan 500 mm. Valmistajan lupaama tarkkuus on 5–10 µm.

Mittalaitteella suoritettiin testejä, joiden ensisijainen tarkoitus oli selvittää mittalaitteen toimintavarmuutta eri mittausmoodeilla ja tutkia laitteen tarkkuutta, kun mitattavan kappaleen sijainti mittausalueella muuttuu. Koska testien pääpaino oli laitteeseen perehtymisellä, pidettiin otantojen määrä testisarjaa kohti melko alhaisena.

Tässä julkaisussa mittausalueella tarkoitetaan nelikulmiota, jonka sivut muodostuvat laserverhon korkeudesta sekä anturien välisestä etäisyydestä (kuva 38).



KUVA 33. *Mittausjärjestely (Outer diameter).*

12.1 TYÖN SUORITUS

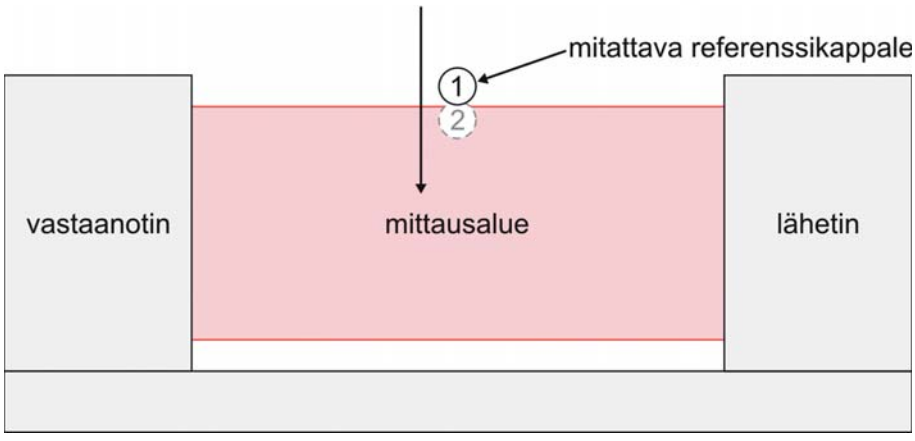
Testejä suoritettiin halkaisija- (Outer diameter) sekä reuna- (Interrupted beam) mittausmoodilla. Nimensä mukaisesti halkaisijamoodi mittaa pyörähdyskappaleen ulkohalkaisijaa ja reunamoodi kappaleen reunan etäisyyttä mittausalueen ylä- tai alareunasta.

Mittaus suoritettiin kiinnittämällä mittalaite koneruuvipuristimella työstökoneen palettiin. Suuntaispaloilla mittalaite asetettiin kohtisuoraan mitattavaa kohdetta vasten, jolloin kulmavirheen vaikutus saatiin minimoitua. Jokaiseen testiin haettiin kokeellisesti mittausalueen ylä- ja alaraja-arvot kuljettamalla refe-

renssikappale mittausalueen reunalle siten, ettei mikrometri saa siitä enää lukemaa. Lisäksi haettiin silmämääräisesti leveyssuunnan raja-arvot viemällä mitattava referenssikappale aivan kiinni vastaanottimeen ja lähettimeen. Tällöin mittausalueen raja-arvot voitiin lukea koordinaattitietoina työstökoneen näytöltä. Koska mittausalueen rajat jouduttiin hakemaan silmämääräisellä tarkkuudella ja mittalaite irrotettiin testien välillä, voidaan yksittäisen testisarjan tuloksia pitää luotettavina. Eri testisarjojen tuloksia ei kuitenkaan voida pitää keskenään vertailukelpoisina. Lisäksi testien nollapiste haettiin melko karkeasti silmämääräisellä tarkkuudella pyöristämällä työstökoneen koordinaatiston arvot tasalukuihin.

Mitattava referenssikappale kiinnitettiin koneen karalle, jolloin työstökoneen näytöltä voitiin lukea millimetrin tuhannesosan tarkkuudella kappaleen asema. Vertaamalla kappaleen asemakoordinaattia mittausalueen raja-arvoihin saadaan kappaleen asema mittausalueella. Havainnoissa esitetyt arvot kappaleen sijainnille ovat valmiiksi laskettuja sijainteja mittalaitteen mittausalueella. Mittausalueen nollapiste sijaitsee vastaanottimen päädyssä mittausalueen alarajalla(Kuva 38) tutkittaessa mittauslukeman muuttumista verhon eri osissa. Reunamittausmoodilla nollapiste valittiin anturien keskelle siten, että referenssikappale oli aivan mittausalueen ylärajan tuntumassa (Kuva 39). Mitattava kappale tuotiin mitattavaksi joka kerta samalta suunnalta, jolloin koneen välykset saatiin myös yhdensuuntaisiksi ja niiden mahdollisesti aiheuttama virhe systemaattiseksi.

Mittaustulokset tallennettiin mittalaitteelta suoraan tietokoneelle Omronin SmartMonitor GT -ohjelmistoa ja RS-232 -liitäntäkaapelia käyttäen. Näin vältyttiin tulosten kirjaamisessa mahdollisesti tapahtuvilta virheiltä.

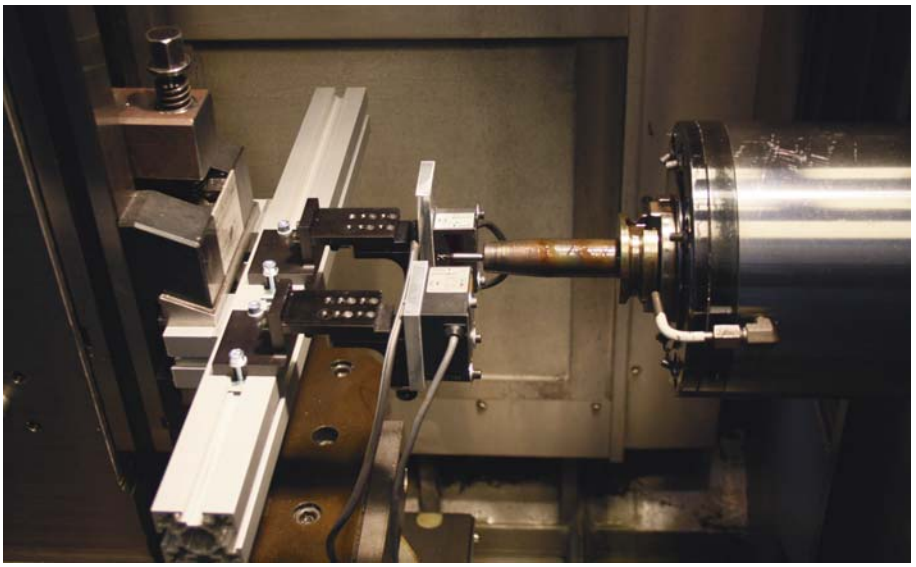


KUVA 34. *Koejärjestely Interrupted beam -moodilla.*

12.2 ANTURIEN KOHDISTAMINEN

Anturien kohdistaminen suoritettiin mittalaitteen valmistajan ohjeiden mukaisesti. Kohdistaminen osoittautui erittäin haastavaksi. Valon intensiteetin ja balanssin arvot huojuivat valmistajan kiinteällä välipalkillakin rajusti. Vaikka anturit olivat paikalleen kiristettynä, heiluivat kohdistuksessa seurattavat arvot valmistajan suosittelemien ohjearvojen ulkopuolella (Tarkat mittaukset: Balanssi alle 10, Intensiteetti yli 130).

Osassa testauksista käytettiin itse tehtyä välipalkkia (kuva 35), jolla kohdistaminen voitiin suorittaa kolmen akselin suunnassa mikrometriruuvisäätöjen avulla. Tällöin balanssi ja intensiteetti saatiin vakaammiksi kuin valmistajan välipalkilla, mutta arvot heiluivat edelleen ohjearvojen ulkopuolella. Kohdistaminen pyrittiin suorittamaan myös valolta eristetyssä tilassa, jolloin ympäristön valaistus ei päässyt vaikuttamaan tulokseen. Arvot eivät kuitenkaan pysyneet vaadituissa rajoissa.



KUVA 35. *Koejärjestely omatekoisella välipalkilla.*

12.3 REFERENSSIKAPPALEET JA MITTAUSOLOSUHTEET

Referenssikappaleina käytettiin 6 mm kovametallitappijyrsimen vartta sekä kalibrointikäyttöön tarkoitettua referenssikappaletta, jonka D= 19,996 mm. Testit ajettiin normaalissa huoneenlämpötilassa. Samalla tavalla järjestettiin myös referenssikappaleiden säilytys. Kappaleen lämpötilan muutoksen arvioitiin olevan suuruusluokkaa ±1 °C. Suuntausvirheen määrittämiseen käytettiin valmistajan koneelle antamaa karan ja paletin suurinta sallittua kulmavirhettä ±1°. Suuntaispalojen avulla kohdistetun mittalaitteen kulmatarkkuudeksi arvioitiin n. ± 1°.

12.4 TULOKSET

TAULUKKO 3. *Testien mittaustulokset.*

Mittaustulokset (mm)							
	Ref. Halkaisija	**Keskiarvo**	**Keskiarvon poikkeama**	**Max**	**Min**	**Max-Min**	**Keski-hajonta**
Testi 1	6,000	6,002	0,002	6,019	5,974	0,045	0,009
Testi 2	6,000	6,009	0,009	6,024	5,983	0,041	0,008
Testi 3	19,996	20,024	0,028	20,035	20,009	0,026	0,006
Testi 5	6,000	5,992	-0,008	6,145	5,913	0,232	0,027
	Ref. Halkaisija	**Ero työstökoneen lukemiin**			**Keskiarvo**	**Max**	**Min**
Testi 4	19,996				0,293	0,309	0,283

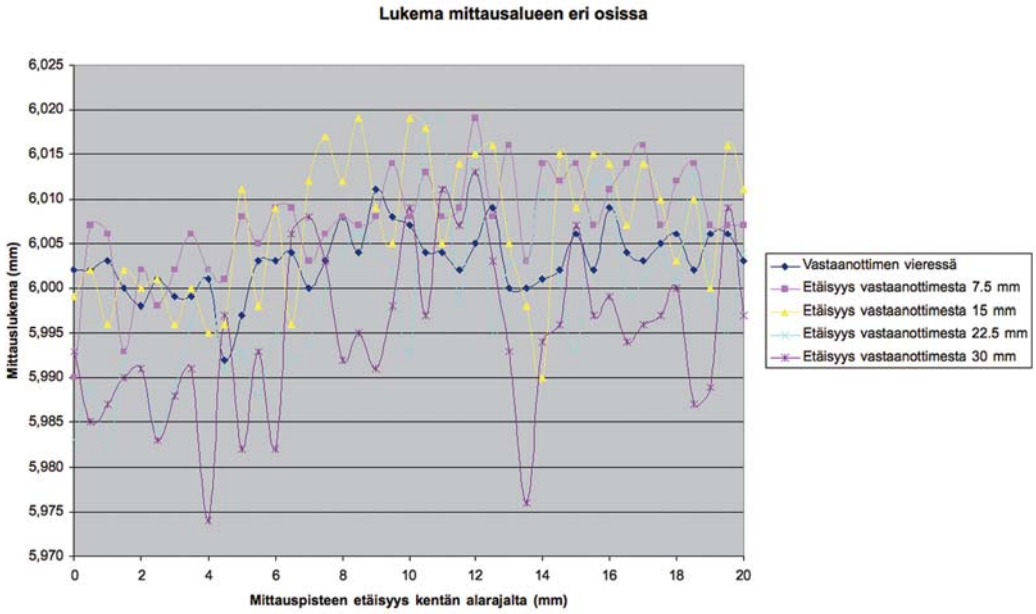
Mittauksissa saadut tulokset pienimmällä anturivälillä ja referenssikappaleella ylittivät vaaditun tarkkuuden (± 10 µm). Kun kappaleen halkaisijaa ja anturiväliä kasvatettiin, heikkeni tulos entisestään. Testitulokset ylittivät eri sarjoilla halutun tarkkuuden niin selkeästi, että useampien toistojen suorittaminen katsottiin tarpeettomaksi.

Katsottaessa tuloksista laadittuja kuvaajia voidaan havaita, että mittausvirheellä ei ole selkeää trendiä. Sen sijaan tulokset heiluvat satunnaisesti puolelta toiselle. Suuremmalla anturivälillä ja 6 mm:n referenssikappaleella (Testi 5) suurimman ja pienimmän arvon välinen ero oli 0,3 mm.

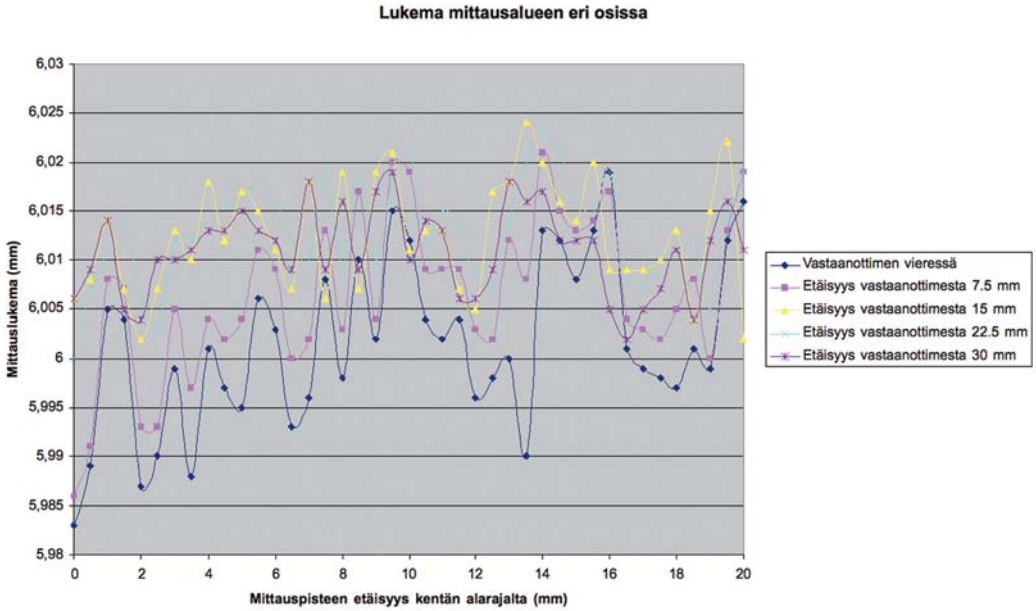
Poikkeuksena tuloksista oli kuitenkin testisarja reunamittausmoodilla, jolloin mittausvirhe esiintyi systemaattisena ja jokaisella mittauksella mittauslukema oli noin 0,3 mm liian suuri.

Kuvioissa 9–13 on esitetty mittalaitteen näyttämä halkaisija nostettaessa kappaletta laservaloverhossa. Jokainen kuvaaja esittää havaintoja ajettaessa mittausalueen alarajalta ylärajalle siten, että etäisyys vastaanottimeen pysyy mittausten aikana vakiona. Mittaukset toistettiin viidellä etäisyydellä vastaanottimesta.

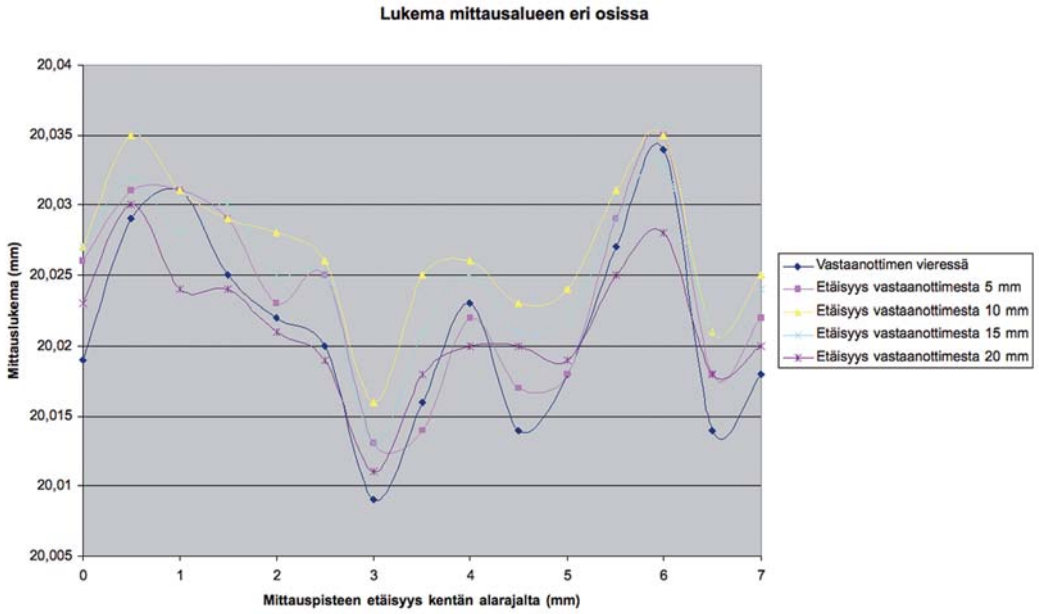
Testi 4. kuvaa reunamoodilla saadun tuloksen poikkeamaa työstökoneelta luettuun siirtymän arvoon.



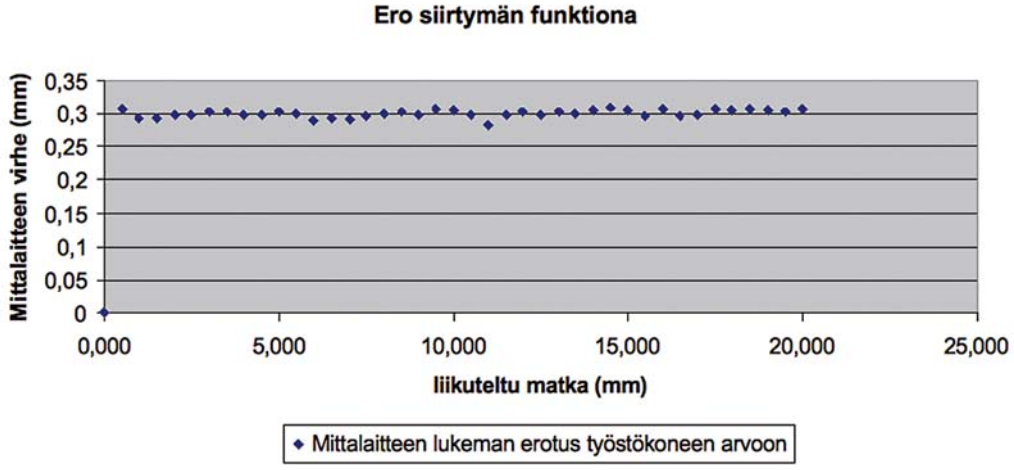
KUVIO 9. *Ensimmäisen testin mittaustulokset.*



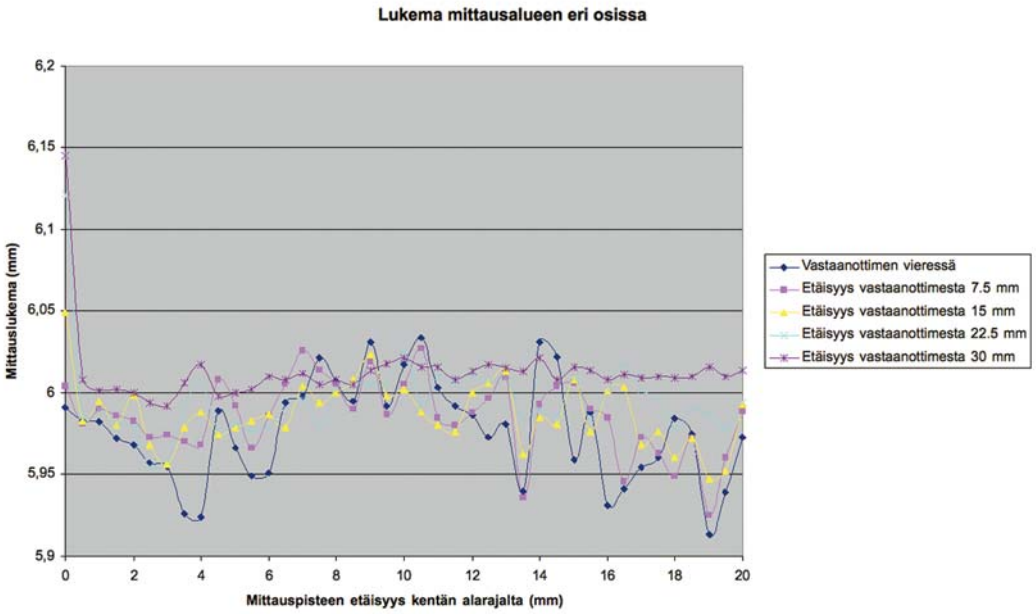
KUVIO 10. *Toisen testin mittaustulokset.*



KUVIO 11. *Kolmannen testin tulokset.*



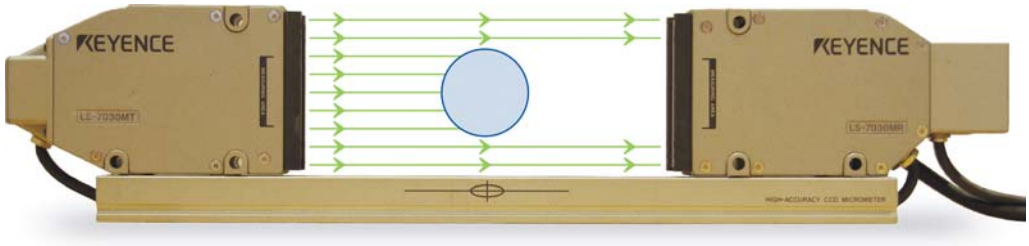
KUVIO 12. *Neljännen testin tulokset.*



KUVIO 13. *Viidennen testin tulokset.*

13 KEYENCE LS-7030M

Pyörähdyskappaleiden ulkohalkaisijan mittaamiseen robottisolussa on optinen mikrometri Keyence LS-7030M ohjausyksiköllä LS-7501. Mittalaite koostuu ohjausyksiköstä, led-lähettimestä ja CCD-vastaanottimesta. Laitteen toimintaperiaate ilmenee kuvasta 46. Lähetinyksikössä on kirkkaat vihreät GaN-ledit, joiden tuottama valo kulkee kollimaattorilinssin läpi. Lähettimestä yhdensuuntaisena tuleva valo kohdistetaan vastaanottimeen, jossa on nopea lineaarinen CCD-kenno. Kun valokenttään viedään kappale, voidaan sen heittämän varjon suuruus laskea CCD-kennon pikseleistä ohjainyksikön signaalinkäsittelypiirin avulla. (Keyence 2001)



KUVA 36. *Keyence LS-7030 -laitteen toimintaperiaate.*

LS-7030M -malli on varustettu CMOS-monitorointikameralla, joka näyttää siluettikuvan mitattavasta alueesta ja sen ympäristöstä. Varsinainen mittaus tehdään hyvin kapealta alueelta, joten mittauksessa käytettävä CCD-kenno lukee vain pientä osaa CMOS-kameran näyttämästä alueesta. Kamera helpottaa kappaleen kohdistamista, kun mitataan esimerkiksi kapeaa uraa pyörähdyskappaleesta. (Keyence 2006, 3.)

Saman ohjausyksikön alle on mahdollista kytkeä kaksi mittalaitetta. Ne voidaan asemoida esimerkiksi mittaamaan suuria halkaisijoita.

TAULUKKO 4. *Keyence LS-7030 tekniset tiedot (Keyence 2006, 8).*

Mittausalue	0,3–30 mm
Pienin tunnistettava kohde	0,3 mm
Lähettimen ja vastaanottimen etäisyys	160±40 mm
Mittaustarkkuus	±2,0 µm
Mittauksen toistuvuus	±0,15 µm
Näytteenottotaajuus	2400 mittausta/s

13.1 TIEDONSIIRTO MITTALAITTEEN JA ROBOTIN VÄLILLÄ, KEYENCE

13.1.1 Vaatimusmäärittely

Ennen rajapinnan ohjelmoinnin aloittamista on tärkeää määritellä, mitä asioita rajapinnan pitää toteuttaa. Vaatimusmäärittelyssä kirjataan vaatimukset ja tavoitteet, joiden toteutuessa ohjelman toiminnallisuutta voidaan pitää valmiina. Nämä osakokonaisuuden tekniset vaatimukset on kirjoitettu vastaamaan työn varsinaisia tavoitteita ja vaatimuksia.

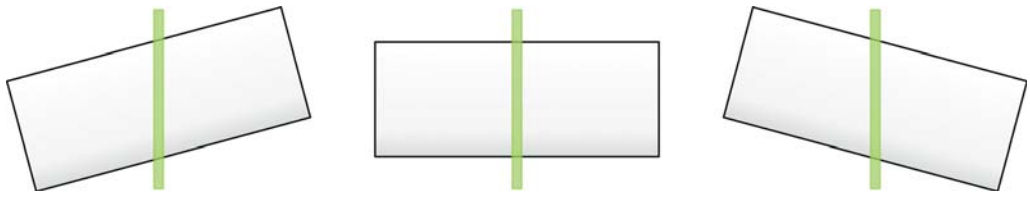
Mittalaitetta ei normaalituotannossa tarvitse käyttää käsin, mutta olemassa oleva käyttöliittymä halutaan silti säilyttää. Laitetta on siis edelleen voitava käyttää myös manuaalisesti.

13.1.2 Mitta-arvon lukeminen

Normaalisti mittalaite näyttää mittaustuloksen näytöllä, josta mittaaja lukee tuloksen. Kehitettävässä sovelluksessa mitta-arvo luketaan sarjaliitännän kautta robotille. Ohjelmalle annetaan ohjelmakutsun yhteydessä parametrina viittaus robotin muistirekisteriin, johon mittauksen tulos palautetaan.

Yksittäisen mitan lukemiseen liittyy haaste: mittaus on altis kosinivirheelle, jos kappale viedään robotilla mittalaitteelle mitattavaksi. Robotin paikoitustarkkuuden rajallisuus aiheuttaa kulmavirhettä. Toinen potentiaalinen lähde kulmavirheelle on kappaleen tartunnan epävarmuus. Tarraimen ja työkappaleen väliin jäänyt lastu tai muu partikkeli asettaa kappaleen vinoon robotin tarraimeen nähden ja aiheuttaa lisävirheen mittaustulokseen.

Kulmavirhe voidaan eliminoida kallistamalla kappaletta mittauksen aikana kuvan 20 mukaan. Tällä kallistusliikkeellä kappale käy jossain vaiheessa oikeassa asennossa ja todellinen halkaisija voidaan lukea koko mittausliikkeen pienimmästä mitta-arvosta.



KUVA 37. *Kulmavirheen eliminointi kappaletta kallistamalla.*

Yksittäisen mittaustuloksen lisäksi pitää siis voida lukea myös liikkeen aikaisen mittauksen minimitulos. Tämä edellyttää kättelysignaalien vaihtoa mittalaiterajapinnan ja robotin työohjelman välillä: mittalaiterajapinnan pitää voida kertoa työohjelmalle, milloin mittausliike voidaan aloittaa ja vastaavasti työohjelman pitää voida kertoa mittalaiterajapinnalle, milloin mittausliike tulee valmiiksi.

13.1.3 Mittausasetusten muuttaminen

Mittalaitteessa on erilaisia käyttöasetuksia, joita osaa pitää voida muuttaa robottiohjelmasta käsin. Muutettavia asetuksia ovat suodatus ja mittausohjelman valinta. Muita asetuksia ei katsottu tarpeelliseksi muuttaa robotilla.

Suodatuksella tarkoitetaan mittalaitteen sisäistä keskiarvoistusta, millä pyritään sisäisen tarkkuuden parantamiseen.

Mittalaitteelle on mahdollista tallentaa 16 erilaista esiasetettua ohjelmaa. Jos on tarpeen määritellä muita asetuksia kuin suodatus, voidaan asetukset syöttää käsin eri ohjelmiin ja vaihtaa mittalaitteen ohjelma robotilta käsin tarpeen mukaan.

13.1.4 Poikkeuksien käsittely

Jos mittalaite ei pysty vastaamaan odotetusti, on ohjelman annettava käyttäjälle virheilmoitus. Tyypillisiä ongelmatilanteita ovat esimerkiksi:

- Mittalaite ei ole kytkettynä päälle.
- Mittalaite on väärässä tilassa.
- Tiedonsiirtokaapeli on irti/vahingoittunut.

13.1.5 Sarjaliikenne

Laitteiden välinen kommunikaatio voi olla rinnakkais- tai sarjamuotoista. Rinnakkaismuotoisessa tiedonsiirrossa bitit kulkevat omia johtimiaan pitkin samanaikaisesti, sarjamuotoisessa samaa johdinta pitkin peräkkäin. Tästä johtuen sarjamuotoinen tiedonsiirto voidaan toteuttaa rinnakkaismuotoista pienemmällä johdinmäärällä ja yksinkertaisemmalla piirilevyllä. (Haltsonen & Rautanen 2008, 149–153.)

Sarjamuotoinen tietoliikenne jaetaan asynkroniseen ja synkroniseen tiedonsiirtoon. Asynkronisessa tiedonsiirrossa ei siirretä kellosignaalia, vaan kummallakin päällä on oma kellonsa. Tämä saattaa aiheuttaa ongelmia, mikäli kommunikoivien laitteiden sarjapiirien kellotaajuudet poikkeavat toisistaan liikaa. Koska erillistä kellosignaalia ei ole, tahdistetaan liikenne alku- ja loppubiteillä. Tämä huonontaa hyötysuhdetta, kun esimerkiksi kahdeksan databitin lähettämiseen tarvitaan lisäksi yksi alku- ja yksi loppubitti. Lisäksi tiedonsiirron varmistamiseen voidaan käyttää pariteettibittiä, jolloin kontrollibittejä on kolme kahdeksaa databittiä kohden.

Synkroninen tiedonsiirto perustuu kiinteään tiedonsiirtotahtiin. Aloitus- ja lopetusbittejä ei käytetä, vaan tietoa siirretään jatkuvasti tietyllä tahdilla eli synkronisesti. Nykyaikaiset nopeat sarjaväylät, kuten tietotekniikassa yleinen USB (universal serial bus) ja esimerkiksi autoteollisuudessa laajasti käytetty CAN (controller area network) ovat luonteeltaan synkronisia. (Koskinen 2006, 31.)

13.1.6 Asynkroninen RS-232 sarjaliitäntä

RS-232 on yleisin asynkroninen sarjamuotoinen liitäntä. Se on tarkoitettu pienille välimatkoille ja suurimmaksi toimintaetäisyydeksi on määritelty 15 metriä. Käytännössä se kuitenkin toimii merkittävästi pidemmilläkin etäisyyksillä. (Koskinen 2006, 30.)

RS-232 -liitynnällä tieto siirretään merkki kerrallaan. Ennen varsinaista dataa lähetetään alkubitti, joka on aina 0. Tämän jälkeen tulevat databitit, joita on tyypillisesti viidestä kahdeksaan. Databittien jälkeen voidaan lähettää pariteettibitti, jolla tarkistetaan lähetetyn datan oikeellisuus. Tiedonsiirron päättävät loppubitit, joita voi olla 1 tai 2. Seuraavan merkin siirto voi alkaa välittömästi loppubitin jälkeen, mutta väylä voi jäädä myös joutotilaan odottamaan seuraavan merkin lähettämistä. Sekä loppu- että joutobitin arvo on looginen 1. (Haltsonen & Rautanen 2008, 153–154.)

Yksinkertaisimmillaan RS-232 -väylä rakentuu kolmesta johtimesta: signaalimaasta, johon viestitasoja verrataan, vastaanottavasta johtimesta ja lähettävästä johtimesta. Vastaanottavasta johtimesta käytetään RS-232 -määrittelyissä merkintää RxD ja lähettävästä johtimesta TxD. Kahden laitteen välisessä kommunikoinnissa vastaanottava ja lähettävä johdin kytketään ristiin: laitteen 1 lähettävä johdin viedään laitteen 2 vastaanottavalle johtimelle ja päinvastoin. Johtoa, jossa laitteiden välille on kytketty signaalimaan lisäksi vain ristikkäiset RxD ja TxD, kutsutaan nollamodeemikaapeliksi. (Koskinen 2006, 30, 264–267.)

Yksinkertaisen kolmijohtimisen tiedosiirron lisäksi RS-232 -määrittelyissä on varattu signaaleja tiedonsiirron ohjausta varten. DSR (Data Set Ready) ja DTR (Data Terminal Ready) -signaalit kertovat, että kumpikin väylän laitteista on toimintavalmiina. CTS (Clear To Send) ja RTS (Request To Send) ovat niin sanottuja vuonohjaussignaaleja. RTS aktivoidaan, kun laite on valmis lähettämään tietoa. Tämän vastaparina on CTS, joka on lähetyslupa. Jos vastaanottopäässä on ruuhkaa ja tiedonsiirtopuskuri tulee täyteen, asetetaan CTS nollatilaan. Tämä kertoo lähettävälle laitteelle, että tietoa ei voida nyt ottaa vastaan. Kun ruuhka vastaanottopäässä on purettu, asetetaan CTS taas 1-tilaan ja tiedonsiirtoa voidaan jatkaa. Nollamodeemikaapelissa vuonohjaus on ohitettu kytkemällä kummankin pään liittimissä ohjaussignaalit yhteen siten, että laite tulkitsee vastapuolen olevan aina valmiina vastaanottamaan tietoa. (Koskinen 2006, 30, 265–266)

13.1.7 Tiedonsiirtoprotokollan kuvaus

Ohjausyksikön käyttöliittymässä voidaan määritellä tiedonsiirtonopeus ja -asetukset. Kommunikointi tapahtuu kaksisuuntaisesti ascii-muodossa ja tietopakettien päätemerkkinä käytetään normaalitilassa rivinvaihtomerkkiä CR. Laite voidaan asettaa myös käyttämään päätemerkkeinä joko CR+LF tai STX+ETX. Laitetason vuonohjausta ei käytetä. (Keyence 2006b, 7–2.)

Kommunikointi tapahtuu aina samalla kaavalla: laitteelle lähetetään ohjausviesti tai kysely, joka päätetään CR-rivinvaihtoon. Laite vastaa määritysten mukaisella viestillä, joka niin ikään päätetään CR-rivinvaihdolla. Käskyissä ja vastauksissa eri lohkot erotellaan pilkulla. (Keyence 2006b, 7–6.)

Esimerkiksi mittatiedon lukeminen tapahtuu lähettämällä viesti Mq,r[CR], missä q on laitteen mittalähdön numero ja r tuloksen tyyppi. Mittalaitteen ohjauksessa voi käyttää kahta eri lähtöä, joilla on eri asetukset. Jos halutaan tietää vain ulkohalkaisija, voidaan käyttää aina lähtöä 1. Tuloksen tyyppi puolestaan voi olla joko 0 tai 1. 0 tarkoittaa pelkän mitta-arvon palauttamista, kun taas 1 palauttaa mitta-arvon lisäksi myös hyväksyntätiedon ohjausyksikölle syötettyjen rajaarvojen mukaan. Laite siis palauttaa haluttaessa myös tiedon siitä, onko kappale toleranssialueella. (Keyence 2006b, 7– 8.)

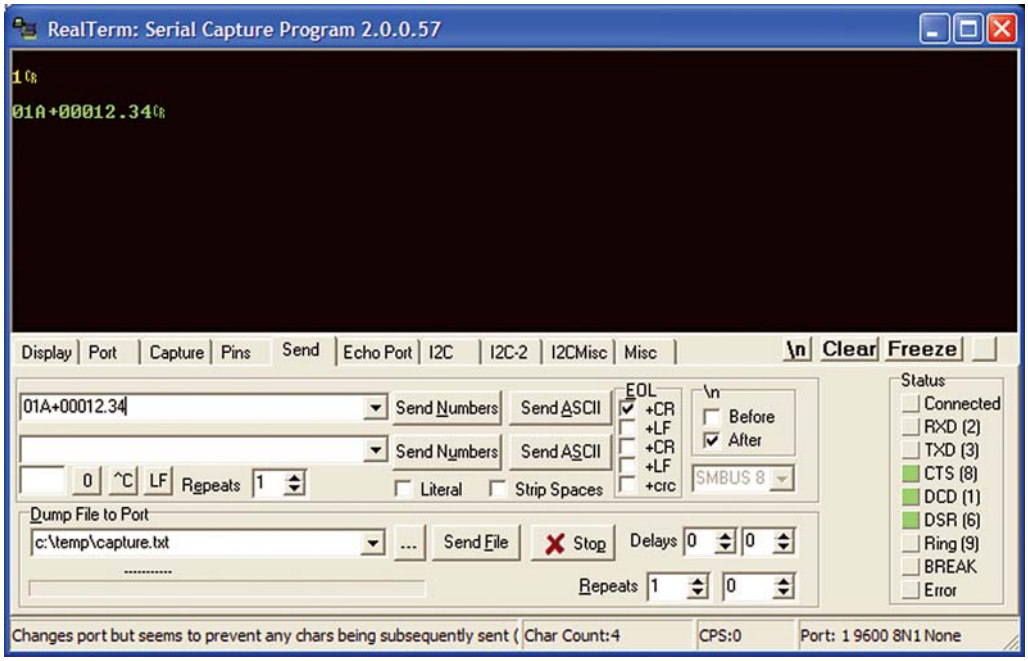
Protokolla on kauttaaltaan melko yksinkertainen ja hyvin dokumentoitu.

13.1.8 Tiedonsiirron testaaminen

Mittalaitteiden sarjaliikenneprotokolla on kuvattu laitetoimittajien teknisessä materiaalissa. Vaikka tekninen määrittely onkin tarkka ja kuvaava, laitetta on kokeiltava käytännössä ennen robotin ja mittalaitteen välisen rajapinnan ohjelmoinnin aloittamista. Helpointa testaaminen on terminaaliohjelman avulla. Mittalaite voidaan kytkeä tietokoneen sarjaporttiin ja terminaaliohjelman avulla voidaan keskustella mittalaitteen kanssa sarjaväylää käyttäen.

Tätä työtä tehtäessä sarjaliikenteen tutkimiseen käytettiin avoimen lähdekoodin terminaaliohjelmaa Realterm, joka soveltuu erittäin hyvin tiedonsiirtoprotokollan testaamiseen ja vianetsintään. Realterm on kehitetty nimenomaan tähän tarkoitukseen. Esimerkiksi ei-näkyvien ohjausmerkkien lukeminen ja lähettäminen onnistuu helposti, mikä on usein tarpeellista kun selvitetään tiedonsiirron toimintaa. Kuvassa 48 näkyy Realtermin kommunikointi-ikkuna, jonka kautta sarjaväylään voi lähettää viestejä. (Realterm 2010.)

Kun tiedonsiirrossa kaikki tarvittavat asiat on testattu ja todettu toimivan odotetusti, voidaan sama toteutus tehdä Karelilla robotin ja mittalaitteen välille. Varsinkin monivaiheisemman kommunikoinnin tapauksessa on järkevää kirjoittaa tiedonsiirron vaiheet ja rakenne suunnittelun avuksi ns. pseudokoodimuotoon ennen varsinaista ohjelmointia.



KUVA 38. *Sarjaliikenteen testaaminen Realterm-ohjelmalla.*

13.1.9 Toteutus

Selkeyden vuoksi kaikki Keyence-mittalaitteen ohjaamiseen liittyvät toiminnot päätettiin yhdistää yhdeksi ohjelmaksi. Mittatiedon lukeminen ja asetusten muuttaminen toimivat samassa ohjelmassa, jonka nimi on Keyence. Tällä ohjelmalla tehdään kaikki Keyence-mittalaitteeseen liittyvät asiat.

Keyence-rajapinta, kuten kaikki muutkin Karel-ohjelmat, koostuu monesta eri osasta ja vaatii huolellista suunnittelua. Koska hyvin suunniteltu on puoliksi ohjelmoitu, tehtiin jokaisesta ohjelmasta yksinkertainen suunnitteludokumentti ennen varsinaista ohjelmointia. Suunnitteludokumentti auttaa ymmärtämään ja suunnittelemaan ohjelman toimintaa tekovaiheessa, mutta toinen yhtä olennainen käyttö sille löytyy, jos ohjelmaa pitää myöhemmässä vaiheessa muokata. Suunnitteludokumentin perusteella ulkopuolinenkin saa nopeasti käsityksen ohjelman toiminnasta ja pystyy tekemään tarvittavat muutokset varsinaiseen lähdekoodiin. Suunnitteludokumentti koostuu kahdesta osasta: käskysyntaksista ja pseudokoodista.

Pseudokoodilla tarkoitetaan luonnollista kieltä muistuttavaa ohjelman rakennetta kuvaavaa tekstiä. Ohjelmoija voi hyödyntää pseudokoodia suunnitteluvaiheessa. Ohjelman toimintalogiikkaa voidaan suunnitella ilman, että tarvitsee kiinnittää tarkkaa huomiota käytettävän ohjelmointikielen käskysyntaksiin. Kirjoitetun pseudokoodin pohjalta on usein melko suoraviivaista kirjoittaa varsinainen ohjelmakoodi tietyn ongelman ratkaisemiseksi. (Ford 2007, 158.)

13.1.10 Virhetilanteista toipuminen

Tyypillinen virhetilanne tai poikkeustilanneon esimerkiksi se, kun ohjelmakutsun parametreissa syötetään tietoa väärässä muodossa tai kun sarjaliikenteessä tulee syystä tai toisesta johtuen tiedonsiirtovirhe.

Karel tarjoaa erilaisia työkaluja poikkeuksien käsittelyyn. Esimerkiksi tiedostojen käsittelyn ja sarjaportin käyttämisen yhteydessä on mahdollista lukea edellisen operaation vikakoodi funktiolla `IO_STATUS(tiedostomuuttuja)`. Tällä voidaan selvittää, onnistuiko aiottu operaatio odotetusti vai aiheutuiko jostain virhe. Sarjaliikenteen onnistumisen tarkkailussa voidaan käyttää tavallista ehtolausetta, missä puskurista luettua dataa verrataan tiedonsiirtoprotokollan määritysten mukaiseen palautusarvoon. Jos puskurista luetaan jotain, mitä sieltä ei pitäisi tulla, keskeytetään ohjelma ja ilmoitetaan käyttäjälle tiedonsiirtovirheestä.

Työn aikana ei löydetty toimivaa keinoa tutkia, onko esimerkiksi käytettävän laitteen tiedonsiirtokaapeli kiinni. Tällöin laite ei vastaa mitään ja ohjelma jää odottamaan portin avaamiseen. Portin avaamisen tilaa ei voida tunnistaa `IO_STATUS`-funktiolla, koska ohjelma jää odottamaan sitä edeltävälle riville. Tämän toiminnallisuuden voisi todennäköisesti toteuttaa moniajon kautta ohjelmallisella vahtikoiralla, joka palauttaisi virheen mikäli portin avaamiseen kuluu liian pitkä aika. Tähän voi myös löytyä suoraviivaisempi ja triviaalimpi ratkaisu, mutta tämän työn puitteissa ongelmaa ei tarkasteltu lähemmin.

Vaikka tiedonsiirtoportin avaamiseen liittyviä virheitä ei pystytä tunnistamaan, voidaan kuitenkin tiedonsiirtoon ja parametrien syöttämiseen liittyvät poikkeukset tunnistaa. Virheen sattuessa ohjelma pitää keskeyttää hallitusti. Karel-ohjelman keskeyttäminen ajon aikana onnistuu ABORT-käskyllä (Fanuc Robotics 2006, 360). Ongelmana tässä on se, että ylätason ohjelmalle, josta käsin Karel-ohjelma on kutsuttu, ei välity tietoa poikkeuksesta. Näin robotin varsinainen työohjelma saattaa jatkaa suoritustaan, eikä käyttäjä saa tietoa siitä, että toimintoa ei voitu suorittaa onnistuneesti loppuun asti.

Käyttäjälle voidaan antaa tiedoksi virheilmoitus tulostamalla tekstiä tperror-virtuaalilaitteeseen, johon tulostettu teksti näkyy ohjainyksikön näytön yläosassa sijaitsevalla tilarivillä. Tperror voi näyttää kerrallaan vain yhden rivin, joten edellinen virheilmoitus tulee ensin poistaa näytöltä. Tämän voi tehdä tulostuksen ohjauskoodeilla. Erikoismerkki 128 tyhjentää tilarivin ja merkki 137 siirtää kursorin rivin alkuun (Fanuc Robotics 2006, 137).

Ongelmallista on se, että käyttäjä ei välttämättä huomaa virheilmoitusta. Näin käy varsinkin, jos robotti jatkaa normaalia työkiertoaan. On tärkeää saada käyttäjälle tieto tapahtuneesta virheestä. Lisäksi on tärkeää keskeyttää robotin työohjelma.

Tämä voidaan toteuttaa esimerkiksi niin, että käytetään erillistä tilarekisteriä, johon ohjelma palauttaa aina poikkeustilanteen sattuessa vikakoodin. Ohjelma-

kutsun jälkeen robotin työohjelmassa jäädään odottamaan, kunnes tilarekisterin arvo on 0. Tällöin robotin työohjelman suoritus keskeytyy, jos Karel-ohjelman ajaminen päätyy virhetilanteeseen. Käyttäjä voi helposti diagnosoida ongelman tilarekisterin avulla. Tämä menettelytapa kuitenkin vaatii robotin työohjelmaan ylimääräisen odotuskäskyn. Olisi luotettavampaa, jos Karel-ohjelma osaisi automaattisesti seisauttaa robotin työohjelman virhetilanteen sattuessa. Näin erillisiä odotuskäskyjä ei tarvittaisi ja tilarekisterin käyttö voitaisiin unohtaa.

Karel-kielessä on käsky PAUSE, joka seisauttaa suoritettavan ohjelman väliaikaisesti. Käskyssä on valinnainen lisämääritys, jolla voidaan valita seisautettava ohjelma. Tällöin syntaksi on PAUSE PROGRAM[n], missä n on ohjelman numero. (Fanuc Robotics 2006, 589).

13.2 MITTAUSTESTIT, KEYENCE LS-7030M

Lähteenä Antti Meriön ja Gaius Voltin projektiraportti.

Mittalaitetta testattiin sekä staattisen että liikkuvan kappaleen mittauk-sessa.

Mittaukset suoritettiin Turun Koneteknologiakeskuksen konesalissa. Staattisten mittausten aikana ilmanlämpötilaa ja kappaleen pintalämpötilaa seurattiin NTC-vastusantureilla. Ilmankosteutta seurattiin hiuskosteusmittarilla.

13.2.1 Tulos, staattinen kappale

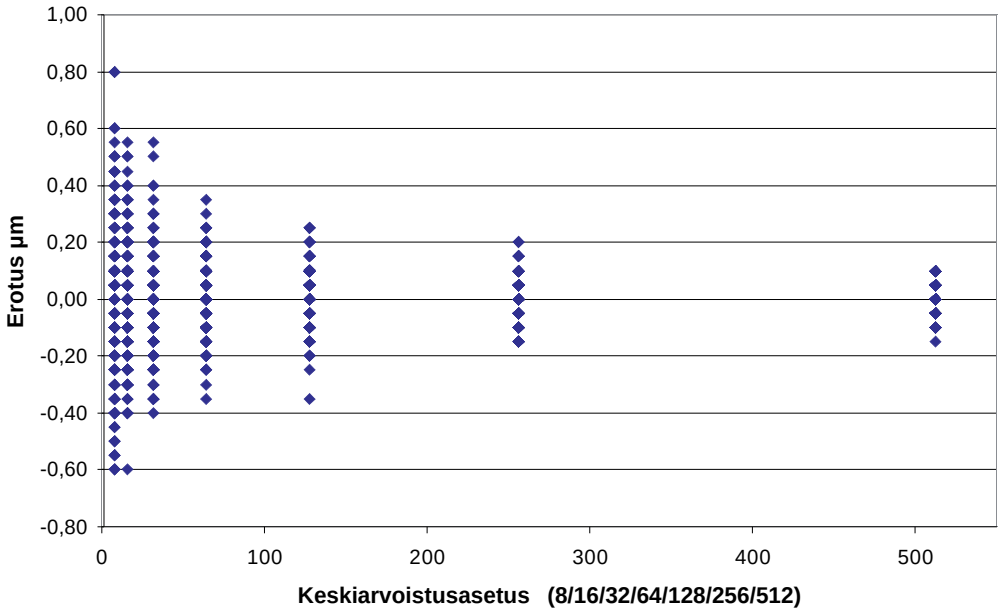
Mittalaitteen sisäiseksi tarkkuudeksi mitattiin ±0,002 mm. Tulos perustuu ylöspäin pyöristettyyn ±3 s hajontaan, jolloin saavutetaan yli 99 % luottamusväli. Tulos on pätevä staattisille kappaleille, mittalaitteen keskiarvoistusasetuksilla 8–1024.

13.2.2 Mittaustapa

Vaikka mittaustulosten laskemisessa käytetään keskiarvoistusta, ei kahden peräkkäisen mittaustuloksen laskennassa ole kuitenkaan käytetty samoja mittauksia. Mittalaitteen suuri mittausnopeus (2400 mittausta/s) takaa sen, että suurimmallakin käytetyllä keskiarvoistuksella (1024) ja minimissään noin 3 sekunnin näytteenottovälillä päällekkäisyyksiä ei tule mukaan näytteeseen. Edellä mainitulla keskiarvoistuksella yhteen mittaustulokseen tarvittavat mittaukset ja laskutoimitukset tapahtuvat noin 0,44 sekunnin aikana. Pienemmillä keskiarvoistusasetuksilla näytteenotto tapahtuu samassa suhteessa lyhyemmässä ajassa. Mittalaitteen dokumentoinnissa laskutoimitusten osuudeksi vasteajasta ilmoitetaan keskiarvoistusasetuksesta riippumatta 2,5 millisekuntia. (Keyence 2001)

13.2.3 Keskiarvoistuksen vaikutuksesta staattisen kappaleen mittauksessa

Vaikka keskiarvoistusasetusta muutetaan radikaalisti, mittalaitteen sisäinen tarkkuus ei kuitenkaan muutu kovinkaan jyrkästi. Muutosta verrattiin tutkimalla saman pisteen mittaustulosta mittalaitteen kahden lähtökanavan välillä. Kanavassa 1 keskiarvoistusasetuksena oli koko mittauksen ajan 1024, ja kanavan 2 keskiarvoistusta muutettiin mittaussuunnitelman mukaisesti.



KUVIO 14. *Keskiarvoistusasetusten vaikutus sisäiseen tarkkuuteen.*

Kuviosta 14 on syytä huomioida, että kunkin keskiarvoistusasetuskohtaisen näytteen mittaustulosmäärä n on 196 kpl ja mittalaitteen erottelukyky 0,05 µm. Tästä johtuen monet pisteet osuvat päällekkäin, eikä kaaviosta voi suoraan nähdä asetuskohtaisten erotusten jakautumaa.

13.2.4 Kosinivirhe

Kosinivirhe on merkittävä mittaustuloksen ulkoiseen tarkkuuteen vaikuttava tekijä. Kosinivirheen aiheuttaa kulmapoikkeama anturilinjan ja kappaleen halutun mittaustason välillä. Mittaustason ja anturilinjan välisen kulmavirheen φ seuraus on kosinivirhe: $D_{tod} = \cos\varphi \cdot D_{mit}$

Optimitilanteessa, kun $\varphi = 0°$, niin $D_{mit} = D_{tod}$.

Staattisen kappaleen mittauksessa kosinivirheen vaikutuksen minimointi on mittaustarkkuuden, etenkin ulkoisen tarkkuuden, kannalta oleellisen tärkeä asia. Liikkuvaa kappaletta mitattaessa kosinivirheen minimointi on helppoa, kunhan liikesuunnaksi voidaan valita sellainen, jossa kappaleen mitattavan tason ja anturilinjan kulma ohittaa riittävän hitaasti pisteen jossa $\varphi = 0°$.

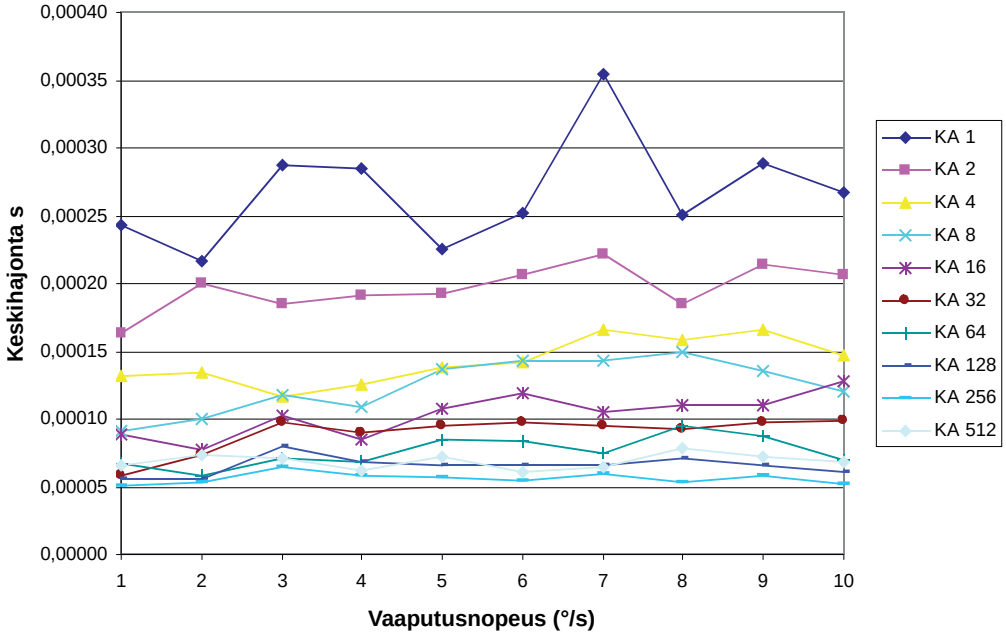
13.2.5 Tulos, liikkuvan kappaleen mittaus

Mittalaitteen sisäiseksi tarkkuudeksi vaaputusliikkeessä olevia kappaleita mitattaessa saatiin ±0,002 mm. Tulos perustuu ylöspäin pyöristettyyn ±3s hajontaan, jolloin saavutetaan yli 99 % luottamusväli. Tulos on pätevä vaaputusliikettä käytettäessä alle Ø 20 mm kappaleelle, mittalaitteen keskiarvoistusasetuksilla 2–512 ja kulmanopeuden ollessa alle 10°/s. Suurempihalkaisijaisilla kappaleilla saavutetaan sama tarkkuus, mutta ensisijaisesti matalammalla keskiarvoistusrajalla ja toiseksi matalammalla vaaputusnopeudella. Esimerkiksi Ø 300 mm kappaleella edellä mainittu tarkkuus saavutetaan keskiarvoistusasetuksilla 2–64 ja kulmanopeuden ollessa edelleen alle 10°/s.

13.2.6 Liikkuvan kappaleen mittaustarkkuus

Testattavan mittalaitteen mittaustarkkuus on riittävä tutkimuksen mukaiseen käyttötarkoitukseen myös liikkuvien kappaleiden mittauksessa. Riittävän tarkkuuden saavuttaminen edellyttää kuitenkin, että mittalaitteen asetukset pitää asettaa mittaustilanteeseen sopiviksi.

Kuviosta 15 voi helposti havaita suuremman keskihajonnan ilman keskiarvoistusta (KA 1), vaaputusnopeudella 7°/s. Suoraa syytä muutokselle ei voitu havaita. Kyseessä saattaa olla esimerkiksi jonkinlainen resonanssi mittauslaitteen ja kappaletta pitelevän robotin välillä. Lähdeaineistona on näyte, jonka mittaustulosmäärä on 100 kpl kutakin keskiarvoistus/vaaputusnopeusyhdistelmää kohti eli yhteensä n on 10000 kpl.



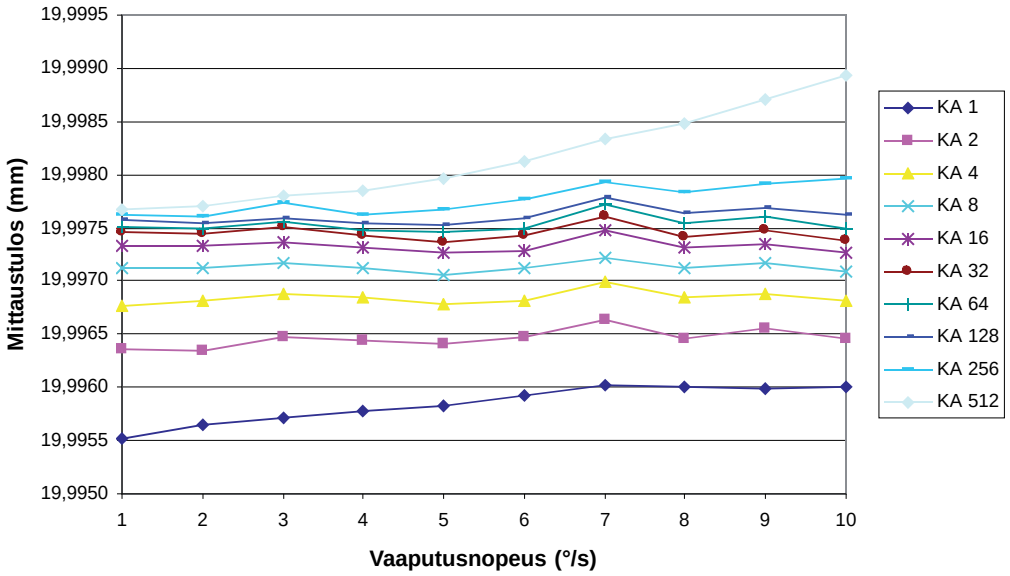
KUVIO 15. *Keskiarvoistuksen ja vaaputusnopeuden vaikutus mittaustulosten keskihajontaan (Ø 20 mm).*

13.2.7 Staattisen ja liikkuvan kappaleen mittauksesta

Testausvaiheessa havaittiin mittalaitteen ulkoisen tarkkuuden paranevan mitattaessa liikkuvaa kappaletta. Staattisen kappaleen mittaustulokset painottuivat todellista mittaa suuremmiksi, mutta liikkuvan kappaleen mittaustuloksissa ei vastaavaa systemaattista virhettä esiintynyt. Syinä tähän ovat osaltaan kosinivirheen minimoituminen vaaputusliikkeen kautta ja mittalaitteen asetukset. Staattinen kappale mitattiin normaali-mittausasetuksella ja liikkuva kappale minimiarvon pito-asetuksella. Minimiarvon pidon voidaan olettaa tilastollisesti pienentävän ylisuurten mittaustulosten esiintymistä.

13.2.8 Keskiarvoistuksen vaikutus mitattaessa liikkuvaa kappaletta

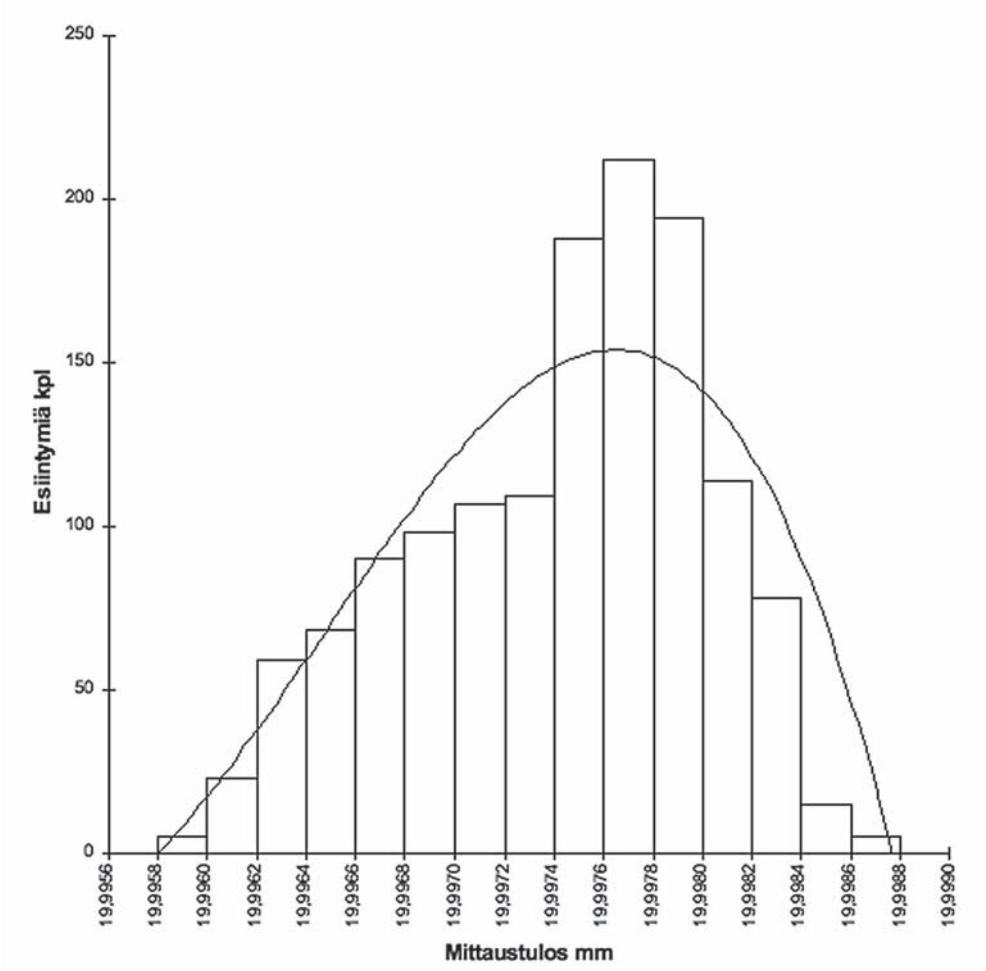
Keskiarvoistusasetuksen nostaminen esimerkiksi asetusta 16 korkeammaksi ei yleensä ole tarpeen, sillä varsinkin suuremmilla halkaisijoilla keskiarvoistuksen nostaminen heikentää mittaustuloksen ulkoista tarkkuutta. Tästä johtuen keskiarvoistusasetuksella 512 aiheutuu merkittävä ulkoisen tarkkuuden heikkeneminen jo niinkin pienellä halkaisijalla kuin 20 mm.



KUVIO 16. *Keskiarvoistuksen ja vaaputusnopeuden vaikutus mittaustulokseen.*

Kuviossa 16 esitettyjen tulosten lähdeaineistona on näyte, jonka mittaustulosmäärä on 100 kpl kutakin keskiarvoistus/vaaputusnopeusyhdistelmää kohti, eli yhteensä n on 10000 kpl (sama aineisto kuin kaaviossa 15).

Tutkittaessa tulosten painottumista (kuva 52) voidaan havaita, että tulokset jakautuvat varsin tasaisesti ja painottuvat kappaleen ilmoitetun nimellismitan tarkkuus huomioiden oikein. Lähdeaineistona on näyte, joka sisältää eri keskiarvoistusasetuksilla samasta kappaleesta tehtyjä mittauksia, yhteensä n on 1365 kpl.



KUVIO 17. *Mittaustulosten jakauma, kun mitattavan kappaleen nimellismitta on 19,998 mm.*

13.2.9 Tarkka mittausalue

Tutkimuksessa havaittiin, että mittalaitteen tarkka mittausalue on käytännössä jonkin verran suurempi kuin valmistaja ilmoittaa. Tarkan alueen todellisia rajoja ei kuitenkaan selvitetty. Rajoihin todennäköisesti vaikuttavat jonkin verran myös laiteyksilökohtaiset erot.

13.2.10 Kotelointi ja suojaus

Mittalaitteen anturiosien kotelointiluokka on IP64, joten laite soveltuu hyvin konepajaolosuhteisiin. Näyttölaitteen näytön kotelointiluokka on IP64, mutta paneelin taakse jäävät liittimet ym. ovat suojaamattomia. Tämä on huomioitava näyttöyksikön sijoituksessa. Myös liitäntäjohtojen liittimien suojaus on heikompi kuin edellä mainittu, joten liittimien paikoitukseen ja johtojen reititykseen on kiinnitettävä normaalia enemmän huomiota niin antureiden, mahdollisten johtojatkosten kuin näytönkin kohdalla.

Mittalaite ei ole erityisen luja mekaanisesti, joten laitteen kiinnityksessä ja koteloinnissa on huomioitava etenkin mittausantureiden suojaaminen mekaanisilta iskuilta. Vaikka isku ei aiheuttaisi näkyviä vaurioita, se voi silti aiheuttaa mittaustarkkuuden heikkenemisen.

14 MITTATIETOJEN KÄSITTELY ROBOTILLA

Lähteenä Sakari Koivusen opinnäytetyö.

14.1 VAATIMUSMÄÄRITTELY

Pelkkä mittaaminen ei vielä ratkaise ongelmia, vaan mittatietoja pitää pystyä hyödyntämään jotenkin. Yksinkertainen tapa hyötyä mittaamisesta on ohjelmallinen tulkki, joka kertoo täyttääkö tuote mittavaatimukset vai ei. Lisäarvoa saadaan mittatietojen kirjaamisella, jolloin asiakkaalle voidaan toimittaa mittauspöytäkirja. Myös mittatietojen kevyt tilastollinen analysointi saattaa olla tarpeellista laadun seurannan ja kehittämisen kannalta.

14.1.1 Hyväksytty/hylätty

Kappale hyväksytään tai hylätään, kun mittatulosta verrataan toleranssirajoihin. Ohjelmalle annettavat parametrit:

- nimellismitta
- toleranssialue
- mittalaitteen epätarkkuus, joka voidaan antaa joko suoraan numeerisesti tai rekisteriviittauksena. Toleranssialuetta pienennetään mittauksen epävarmuuden verran kummastakin suunnasta, jolloin saadaan luotettava tulos.
- rekisteri, johon hyväksytty/hylätty -tulos palautetaan.

Ohjelma palauttaa totuusarvon hyväksytty/hylätty. Tämä tallennetaan vapaasti valittavaan rekisteriin. Rekisteriarvon perusteella voidaan robotin työohjelmassa esimerkiksi viedä hylätyt kappaleet eri paikkaan kuin hyväksytyt, sytyttää merkkivalo tai kirjata kappaleen hylkäys mittauspöytäkirjaan.

14.1.2 Mittauspöytäkirja

Yksi robotisoidun tuotannon eduista on, että robotti voidaan ohjelmoida kirjaamaan erilaisia tietoja määrävälein, esimerkiksi jokaisen valmistuneen kappaleen kohdalla. Tällaisia tietoja voivat olla esimerkiksi tahtiaika, tuotannossa tulleet häiriöt ja kappaleiden mittaustulokset.

Tiedon kirjaamiseen on vaihtelevia tarpeita. Tässä työssä tarvitaan kappaleiden halkaisijamittojen kirjaamista, mutta tietoja kirjaavasta ohjelmasta on tarpeen tehdä joustava siten, että sitä voidaan käyttää ilman muutoksia mihin tahansa tiedonkeruuseen mahdollisimman laajasti.

Kappaleiden mittaustuloksista koostetaan pöytäkirja, jossa voidaan kirjata esimerkiksi seuraavia asioita:

- päivämäärä
- kellonaika
- mittaustulos
- kappaleen tunnus (piirustusnumero)
- kappaleen yksilöllinen tunnus
- mitattava kohta
- hylkäystieto
- tämän kappaleen jälkeen annettu teräkorjaimen muutosarvo.

Ylläolevat tiedot kirjaava ohjelma sopii vain yhteen käyttöön. Ohjelman pitää olla mahdollisimman joustava, joten sen rakenteesta tulee dynaaminen. Ohjelmalle voi antaa parametrina vapaasti valittavia tietoja, jotka kirjoitetaan tiedostoon. Tiedot pitää voida tuoda helposti esimerkiksi taulukkolaskentaohjelmaan mahdollista jatkokäsittelyä varten.

Tiedot kirjaavan ohjelman pitää täyttää seuraavat vaatimukset:

- Se kirjoittaa halutut tiedot tiedostoon.
- Tiedoston nimi on vapaasti valittavissa.
- Tiedosto luodaan, jos sitä ei ole vielä olemassa.
- Päivämäärä/aikaleima -kenttä on valinnainen ja se voidaan ottaa mukaan tai jättää pois ohjelmakutsun parametrilla
- Muita kenttiä voi olla vapaasti valittava määrä (ohjauksen asettamissa rajoissa).
- Kenttien tiedot voivat sisältää joko tekstiä tai numeroita.
- Kenttien tiedot voi syöttää käsin tai ne voivat tulla rekistereistä/tuloista/lähdöistä.
- Tallennuskohde on valittavissa ohjelmakutsun yhteydessä (USB-muisti tai CF-muistikortti).

14.1.3 Tilastollinen analysointi

Tilastollinen analysointi on yleensä melko yksinkertaista normaalissa konepajamittauksessa. Muutamalla perusoperaatiolla pystytään suoriutumaan jo monesta tehtävästä:

- edellisen tai n:nen kappaleen mittaustulos
- aritmeettinen keskiarvo n edellisestä kappaleesta

- keskihajonta n edellisestä kappaleesta
- n edellisen kappaleen minimiarvo
- n edellisen kappaleen maksimiarvo.

Ohjelman pitää palauttaa mikä tahansa näistä arvoista haluttuun rekisteriin. Lisäksi ohjelman pitää tuottaa yhteenvetonäkymä, mihin voidaan tulostaa esimerkiksi 50 edellisen mittauksen tilastolliset tunnusluvut.

14.2 TULKIN TOTEUTUS

Ohjelmallinen tulkki on hyvin suoraviivainen. Parametreina annetaan nimellismitta, toleranssialueen alaraja, toleranssialueen yläraja, rekisteri johon tulos syötetään sekä mittalaitteen epävarmuus. Epävarmuus on valinnainen parametri, jota ei ole pakko syöttää. Mittalaitteen epävarmuus on kuitenkin hyvä ottaa huomioon, jotta vältytään virheellisten kappaleiden hyväksymiseltä.

Koska mittatiedot voidaan syöttää joko kokonais- tai liukulukuina, pitää ohjelman tunnistaa tietotyyppi ja kääntää kokonaisluku liukuluvuksi. Tämä tehdään parametrien mitta, alaraja, yläraja ja virhe kohdalla.

Ohjelma palauttaa tulosrekisteriin arvon 1, jos mitta on toleranssialueella ja arvon -1 jos mitta on toleranssialueen ulkopuolella.

Tulkin suunnitteludokumentti:

```
TULKKI(mitta, alaraja, yläraja, tulosrekisteri, [mitta-
laitteen epävarmuus])

pseudokoodi

pääohjelma
    lue parametri 1 (mitta)
      jos ei löydy, palauta virhe
      jos on väärää tyyppiä (ei liukuluku eikä kokonaisluku),
      palauta virhe
      jos on kokonaisluku, muuta liukuluvuksi
      tallenna oikea arvo mitta-muuttujaan
    lue parametri 2 (alaraja)
      jos ei löydy, palauta virhe
      jos on väärää tyyppiä (ei liukuluku eikä kokonaisluku),
      palauta virhe
      jos on kokonaisluku, muuta liukuluvuksi
      tallenna oikea arvo alaraja-muuttujaan
```

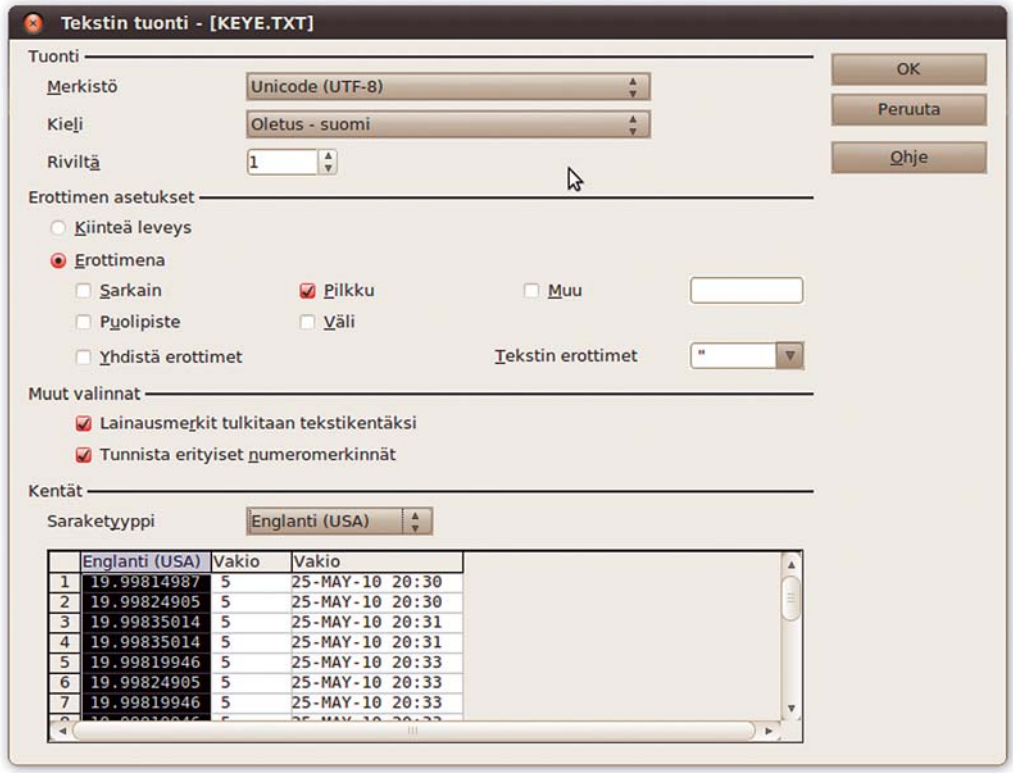
```
    lue parametri 3 (yläraja)
      jos ei löydy, palauta virhe
      jos on väärää tyyppiä (ei liukuluku eikä kokonaisluku),
      palauta virhe
      jos on kokonaisluku, muuta liukuluvuksi
      tallenna oikea arvo ylaraja-muuttujaan
    lue parametri 4 (tulosrekisteri)
      jos ei löydy, palauta virhe
      jos on väärää tyyppiä (ei kokonaisluku),
      palauta virhe
      tallenna oikea arvo tulosrek-muuttujaan
    lue parametri 5 (mittalaitteen epävarmuus)
      jos ei löydy, aseta epavarmuus=0 ja jatka ohjelmaa
      jos on väärää tyyppiä (ei liukuluku eikä kokonaisluku),
      palauta virhe
      jos on kokonaisluku, muuta liukuluvuksi
      tallenna oikea arvo epavarmuus-muuttujaan
      jos mitta - epavarmuus < alaraja TAI mitta +
      epavarmuus > ylaraja, palauta tulosrekisteriin -1
      muuten palauta tulosrekisteriin 1
ohjelman loppu
```

14.3 MITTAUSPÖYTÄKIRJAN TOTEUTUS

Tiedot pitää pystyä kirjaamaan formaatissa, joka on helposti jatkokäytettävissä. Tähän valittiin yleisesti käytössä oleva CSV (comma separated values) -formaatti, jossa tietokentät on erotettu toisistaan pilkuilla. CSV-muotoiltu tiedosto aukeaa tyypillisesti esimerkiksi taulukkolaskentaohjelmissa. OpenOffice.orgin taulukkolaskentaohjelman CSV-tuonti-ikkuna on esitetty kuvassa 53.

Ohjelmalle voi antaa halutun määrän parametreja, jotka kirjoitetaan CSV-tiedoston tietokenttiin. Tiedot voivat sisältää tekstiä, numeroita, rekisteriarvoja tai esimerkiksi tulojen tai lähtöjen tilatietoja. Mittatiedon pitää olla ensimmäinen kirjoitettava parametri myöhemmin tapahtuvan tietojen analysoinnin vuoksi.

Jokaisen rivin perään kirjoitetaan lisäksi päivämäärä ja kellonaika.



KUVA 39. *CSV-tiedoston tuonti taulukkolaskentaan.*

14.3.1 Poikkeustilanteiden hallinta

Kun tietoja kirjataan muistiin, on tärkeää varmistua, että muistilaite on kytkettynä robottiin ja valmis ottamaan tietoa vastaan. Muuten voi käydä niin, että mitataan tuotteita ja säädetään teräkorjaimia mittauspöytäkirjan historiatietojen perusteella, mutta historiatiedot eivät pidä paikkaansa, jos muistilaitteelle kirjoittaminen ei ole onnistunut. On välttämätöntä tunnistaa poikkeustilanne, pysäyttää ohjelman suorittaminen ja varoittaa käyttäjää, jos muistilaitteelle kirjoittaminen ei jostain syystä onnistu.

Tyypillinen ongelma muistilaitteen käyttämisessä on yksinkertaisesti se, että esimerkiksi USB-muistitikkua ei ole liitetty robotin ohjaimeen. Toinen ongelmatilanne tulee silloin, kun muistilaitteen tallennuskapasiteetti tulee täyteen.

`IO_STATUS`-funktio palauttaa edellisen tiedosto-operaation tilatiedot. Sen tuloste kertoo, onnistuiko operaatio ongelmitta. Jos operaatio ei onnistunut, funktio palauttaa vikakoodin. Virhekoodi 12327 viittaa ongelmaan tiedoston avaamisessa. Muistilaitteen valmiutta päätettiin testata siten, että heti tiedoston avaamisen jälkeen luetaan `IO_STATUS` ja jos se poikkeaa nollasta, keskeytetään ohjelma ja palautetaan käyttäjälle virheilmoitus. Tämä ei kuitenkaan toiminut. Tiedoston avaamisen `IO_STATUS` on testien perusteella on aina 0, vaikka muistilaite ei olisikaan paikoillaan. (Fanuc Robotics 2006, A-204).

Koska tämä varmistus on tietojen kirjaamisen kannalta hyvin tärkeä, se piti ratkaista jotenkin toisin. Päätettiin käyttää hyvin yksinkertaista ja varmaa tapaa todeta, onnistuuko tiedostoon kirjoittaminen: avataan testitiedosto, kirjoitetaan sinne jotain, suljetaan tiedosto, avataan se uudelleen ja verrataan luettua sisältöä siihen, mitä tiedostoon juuri kirjoitettiin. Jos tämä ketju menee läpi ongelmitta, muistilaite on käytettävissä.

Muistilaitteen tarkistamiseen liittyvä toiminnallisuus kapseloitiin funktioksi kohde_valmis, joka palauttaa boolen totuusarvon. Jos funktion palautteena on true, tiedetään kohdemuistin olevan valmiina ja käyttökunnossa. Tätä tarkistusta käytetään ohjelmissa `kirjaa` ja `analysoi`.

Käytännössä funktion toiminnallisuus poikkeaa hieman yllä kuvatusta. Jos tehdään suoraviivaisesti niin, että kirjoitetaan tiedostoon, luetaan tiedostosta ja verrataan näitä kahta keskenään, törmätään ei-toivottuun virheeseen. Kun muistilaite ei ole valmiina, sekä tiedostoon kirjoittaminen että tiedostosta lukeminen epäonnistuvat. Tästä johtuen muuttuja, johon tiedoston sisältöä luetaan, jää alustamattomaksi. Kun alustamattoman muuttujan sisältöä verrataan tiedostoon kirjoitettuun tekstiin, ohjelman suoritus keskeytyy ajonaikaiseen virheeseen alustamattoman muuttujan takia.

Tämä ongelma voidaan kääntää ominaisuudeksi, jonka avulla tarkistus on helppo toteuttaa. Ei verrata tiedoston sisältöä varsinaisesti mihinkään, vaan tyydytään yksinkertaisempaan vertailuun: jos muuttuja on alustamaton, tiedetään tiedostosta lukemisen epäonnistuneen. Tämä johtuu suurella todennäköisyydellä juuri siitä, että muistilaite ei ole käytettävissä. Funktion toimintalogiikka käy selkeästi ilmi alla olevasta pseudokoodista:

```
aliohjelma kohde_valmis
    avaa tiedosto
    kirjoita tiedostoon jotain
    sulje tiedosto
    avaa tiedosto uudelleen
    lue tiedostosta kaksi merkkiä muuttujaan testi
    jos muuttuja testi on alustamaton
      palauta false
      sulje tiedosto
    muuten
      palauta true
      sulje tiedosto
```

14.3.2 Parametrien lukeminen

Ohjelmakutsussa voidaan antaa vapaasti valittava määrä parametreja. Ainoan rajan asettaa se, että ohjelmakutsussa on mahdollista antaa vain 10 parametria (Fanuc robotics a, 1075). Ensimmäinen parametri on mittatieto, mutta sen jälkeen voi tulla mitä tahansa tietoa. Ohjelma lukee parametrit 3 - 10 toistosilmukassa, josta poistutaan, kun luettavaa parametria ei löydy.

Koska parametreissa voi olla mitä tahansa tietoa, pitää tietotyyppi ensin tulkita ja sen jälkeen kaikki tiedot muutetaan merkkijono-tyyppiseksi. Merkkijonoksi muutettu tieto lisätään muuttujaan, joka ohjelman päättyessä kirjataan tiedostoon uudeksi riviksi.

14.3.3 Suunnitteludokumentti

```
KIRJAA(tiedosto, muistilaite, tieto1, tieto2, ..., tieto
10)

tiedosto: tiedoston nimi tunnisteineen, esimerkiksi ”tes-
ti.txt”

muistilaite: 1=CF, 2=USB

tieto1...: vapaasti valittava tieto


pseudokoodi

aliohjelma kohde_valmis
    avaa muistilaitteelta tiedost0 testitiedosto.tst
    kirjoita tiedostoon ”OK..”
    sulje tiedosto
    lue tiedostosta kaksi merkkiä muuttujaan
    jos muuttuja ei ole alustettu, palauta FALSE
    muutoin palauta TRUE

lue parametri 1 (tiedostonnimi)
    jos ei löydy, palauta virhe
    jos on väärää tyyppiä (ei kokonaisluku), palauta virhe
    tallenna nimi tiedosto-muuttujaan
lue parametri 2 (muistilaite)
    jos ei löydy, palauta virhe
    jos on väärää tyyppiä (ei kokonaisluku), palauta virhe
    tallenna oikea arvo muistilaite-muuttujaan
```

```
lue parametrit 3...10
    jos parametria ei löydy, poistu silmukasta
    jos ei ole merkkijono, muuta merkkijonoksi
    lisää parametrin sisältö ja pilkku kirjaus-muuttujaan

kun kaikki parametrit on luettu, kirjoita ajankohta kirja-
us-muuttujan loppuun
avaa tiedosto
anna virhe jos tiedostoa ei voi käsitellä (tikku irti tms.)
kirjoita uusi rivi tiedostoon (muuttujan kirjaus sisältö)
sulje tiedosto
```

14.4 TIETOJEN ANALYSOINNIN TOTEUTUS

Tietojen analysointiohjelma lukee CSV-tiedostoa, johon on tallennettu mittatietoja. Koska ohjelma on hieman laajempi kuin muut tässä työssä toteutetut ohjelmat, sen suunnittelu jaettiin kahteen osaan: tietojen analysointi ja tietojen lukeminen tiedostosta.

14.4.1 Suunnitteludokumentti, aliohjelmat analysointiin

Mittatiedostot on tallennettu CSV-muotoiseen tekstitiedostoon. Analysointiohjelman tulee lukea haluttua tiedostoa, erotella sieltä mittaustiedot ja laskea tiedoista haluttu tilastollinen tunnusluku. Ohjelman voidaan ajatella koostuvan kahdesta osasta: mittatietojen erottelutiedostosta ja toisaalta tietojen analysoinnista. Ohjelman rakennetta alettiin suunnitella pseudokoodin avulla. Ensimmäisessä vaiheessa huomiota ei kiinnitetty tietojen lukemiseen, vaan pyrittiin toteuttamaan tarvittavat analysointifunktiot.

Koska tiedostot voivat sijaita eri muistilaitteilla ja mittaustiedostoja voi olla useita, ohjelmalle pitää voida antaa parametrina tiedoston nimi ja sijainti.

```
ANALYSOI(tiedosto, muistilaite, toiminto, määrä, tulosre-
kisteri)
tiedosto: tiedoston nimi tunnisteineen, esimerkiksi ”tes-
ti.txt”
muistilaite: 1=CF, 2=USB
toiminto:
```

```
    - arvo (palauttaa esimerkiksi kolmanneksen tuoreimman
    arvon)
    - keskiarvo (palauttaa n. kappaleen aritmeettisen
    keskiarvon)
    - keskihajonta (palauttaa n. kappaleen keskihajonnan)
    - min (palauttaa n kappaleen minimiarvon)
    - maks (palauttaa n kappaleen maksimiarvon)
    - yhteenveto
määrä: miten monen kappaleen otanta/miten mones kappale
tulosrekisteri: mihin rekisteriin tulos tallennetaan?

pseudokoodi

aliohjelma keskiarvo
    taulukosta n mittaustulosta
      summataan tulokset yhteen
    jaetaan n:llä
    palauta tulos
aliohjelma minimi
    välitulos = taulukon ensimmäinen mittaustulos
    taulukosta loput mittaustulokset
      jos uusi tulos on pienempi kuin välitulos, muutetaan
välituloksen arvoksi uusi tulos
    palauta tulos
aliohjelma maksimi
    välitulos = taulukon ensimmäinen mittaustulos
    taulukosta loput mittaustulokset
      jos uusi tulos on suurempi kuin välitulos, muutetaan
välituloksen arvoksi uusi tulos
    palauta tulos
aliohjelma hajonta
    välitulos = 0
    käytetään keskiarvo-aliohjelman tulostetta, tallenne-
taan muuttujaan ka
    taulukosta n mittaustulosta
      välitulos = välitulos + (mittaustulos - ka)^2
    palauta neliöjuuri(välitulos/määrä)
```

Tietojen analysointi on melko suoraviivaista. Eri toimintoja toteuttavat aliohjelmat oli hyvin suoraviivaista toteuttaa pseudokoodin pohjalta. Oikea aliohjelma kutsutaan ohjelmakutsussa annettavan parametrin perusteella.

14.4.2 Tietojen lukeminen

Tiedot on tallennettu tiedostoon pilkuilla erotettuihin kenttiin. Mittausdata on aina ensimmäisessä sarakkeessa, joten analysointiohjelman pitää lukea jokaiselta riviltä tietoa ensimmäiseen pilkkuun saakka.

Kun tiedoston rivi on luettu ensimmäiseen pilkkuun asti, muutetaan luettu data liukuluvuksi. Tiedostossa voi olla jotain muutakin kuin validia numerotietoa, joten muunnoksen onnistuminen pitää varmistaa. Epäonnistuneen muunnoksen jälkeen jatketaan tiedoston lukemista rivin loppuun saakka ja aloitetaan seuraavan rivin alussa olevan arvon lukeminen.

Merkkijonon muuttaminen liukuluvuksi onnistuu sisäänrakennetulla funktiolla `CNV_STR_REAL(lähde,kohde)`. Jos muunnos ei onnistu, funktio palauttaa kohdemuuttujan alustamattomana, jolloin siihen ei ole liitetty mitään arvoa. (Fanuc Robotics 2006, 412.) UNINIT-funktiolla on mahdollista tarkastaa, onko muuttujalla arvoa vai ei (Fanuc Robotics 2006, 691). Sitä käytetään tietokentän oikeellisuuden tarkistamiseen.

14.4.3 Tietojen säilyttäminen taulukossa

Tiedostosta luettu data pitää säilyttää ohjelman suorituksen ajan muistissa analysointia varten. Muuttujataulukko on tähän luonteva ratkaisu. Alussa ohjelman suunniteltiin toimivan niin, että muuttujataulukkoa kasvatetaan aina jokaisen onnistuneesti luetun mitta-arvon kohdalla, mutta koska Karelissa ei ole mahdollista muuttaa taulukon kokoa ohjelman suorituksen aikana, suunnitelmia piti muuttaa (Fanuc Robotics 2006, 2–20).

Tietojen taulukointi toteutettiin siten, että taulukolla on kiinteä pituus. Pituudeksi valittiin 100 alkiota. Kun yritetään lisätä alkiota 101, taulukkoa kierrätetään niin, että vanhin arvo putoaa pois. Tämä tehdään silmukalla, missä käydään alkion arvot läpi: alkio 1 saa alkion 2 sisällön, alkio 2 saa alkion 3 sisällön jne. kunnes uusin arvo lopulta tallennetaan alkioon 100. Alkiossa 1 on aina vanhin arvo.

Lisäksi tässä menettelyssä on pidettävä kirjaa siitä, miten monta alkiota taulukosta on käytetty. Taulukossa on 100 alkiota, vaikka aktiivisessa käytössä niitä olisi vähemmän.

14.4.4 Tietojen lukeminen, pseudokoodi

Kun analysointia varten tehdyt aliohjelmat toimivat, kirjoitettiin testien ja suunnittelun pohjalta pseudokoodi myös tiedoston lukemista ja tietojen taulukointia varten.

```
aliohjelma lisaa (lisätään alkio taulukkoon)
    jos alkiot_lkm<taulukon pituus (testataan mahtuuko taulukon perään uusi arvo)
      taulukko[alkiot_lkm]=uusi arvo
      lisätään laskurimuuttujaa alkiot_lkm yhdellä
    muutoin (taulukko on jo täynnä)
      tehdään tilaa uudelle arvolle, siirretään jokaista taulukon arvoa ja pudotetaan vanhin (1) pois
      toista 2...taulukon pituus
        taulukko[i-1]=taulukko[i]
      taulukko[taulukon pituus]=lisättävä arvo

aliohjelma lue_tiedot
    avaa tiedosto, virhe jos ei löydy
    nollaa muuttuja alkiot
    toista kunnes tiedosto on lopussa (luetaan kaikki rivit)
      nollaa rivi-muuttuja
      toista kunnes ollaan pilkussa tai rivinvaihdossa (luetaan pilkkuun saakka)
        lue yksi merkki
        jos tultiin tiedoston loppuun, mene aliohjelman loppuun
      jos merkki ei ole tyhje tai pilkku, lisää se rivi-muuttujaan
      toista kunnes ollaan rivinvaihdossa (luetaan rivi loppuun)
        lue yksi merkki
        jos tultiin tiedoston loppuun, mene aliohjelman loppuun
      rivi luettu, kokeillaan kääntää rivi-muuttuja reaali-luvuksi
      jos onnistuu, lisätään arvo taulukkoon lisaa-
      aliohjelmalla

    lue rivit taulukkoon
    suorita oikea aliohjelma parametrien perusteella

    palauta tulos rekisteriin
```

15 TYÖKALUKORJAIMEN SIIRTO ROBOTILTA MONITOIMISORVILLE

Sorvin terägeometriaan voidaan asettaa korjaimia, joilla kompensoidaan esimerkiksi teräpalojen kulumista. Perinteisesti sorvia käyttävä koneistaja syöttää terän korjaimen arvon manuaalisesti mittaustulosten perusteella. Tässä työssä tarkastellaan mittalaitteen ja robotin välisen rajapinnan toteuttamista, joten manuaalinen mittaus jätetään pois ja se korvautuu robotin suorittamalla mittauksella.

Mittatiedon perusteella robotti voi hyväksyä tai hylätä koneistetut kappaleet, mutta oletusarvoisesti robotti ei anna sorville mitään takaisinkytkentää mittatiedoista. Pelkän mittatarkkuuden tarkastamisen lisäksi prosessia halutaan myös säätää, jotta hylättäviltä tuotteilta voidaan välttyä tai niiden määrä voidaan minimoida. On tärkeää saada siirrettyä sorville takaisinkytkentätieto mittapoikkeamasta, jos kappaleiden mitat alkavat vaeltaa kohti toleranssialueen ääripäätä.

15.1 TYÖKALUN KÄSITTELY

Numeerisesti ohjatuissa työstökoneissa on tyypillisesti useita työkaluja, jotka voidaan vaihtaa automaattisesti ohjelmakierron aikana (Ansaharju & Maaranen 2000, 463). Työkalun vaihto suoritetaan M-koodilla M6. T-koodeilla otetaan valittu työkalu käyttöön ja liitetään työkaluun halutut korjainarvot. T-koodissa on NC-sorveissa tyypillisesti neljä numeroa osoitteen T perässä. Kaksi ensimmäistä numeroa kertovat työkalun numeron eli revolveripaikan ja kaksi viimeistä muistipaikan, josta työkalukorjain haetaan. Usein on selkeintä, että työkalun ja korjaimen numero pidetään samana. (Ansaharju & Maaranen 2000, 491–493).

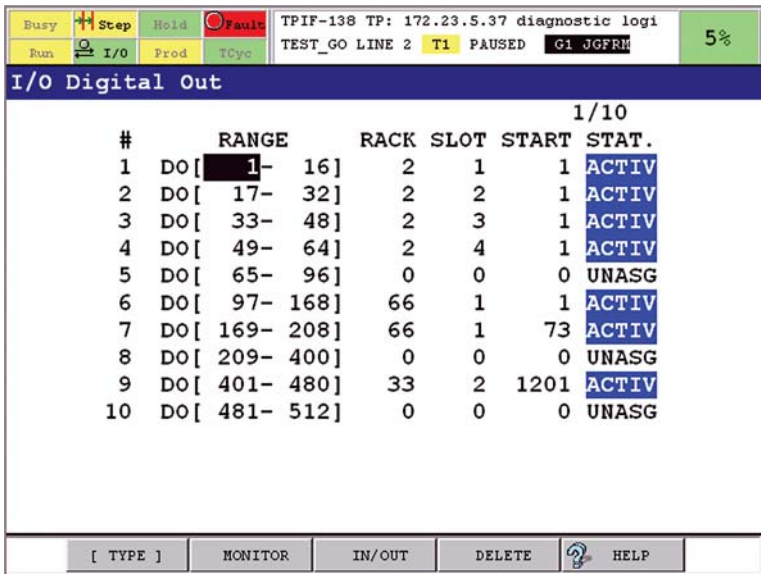
Fanuc Series 18i-TB -ohjauksessa työkalukorjain on jaettu kahteen osaan: geometrian korjaimeen ja kulumisen korjaimeen. Geometrian korjaimella annetaan tiedot työkalun mitoista ja terän nirkon alkuperäisestä sijainnista. Kulumisen korjaimella kompensoidaan nimensä mukaisesti teräpalan työstönaikaista kulumista, mutta myös koneen lämpötilan vaihteluista ja muista ympäristötekijöistä johtuvia muutoksia. (Fanuc Ltd. a 2001, 220–224).

15.2 ROBOTIN JA SORVIN VÄLINEN TIEDONSIIRTO

Robotti ja sorvi on kytketty toisiinsa Profibus-kenttäväylän avulla. Väylää pitkin kulkevat esimerkiksi robotin lähettämä käynnistyskäsky sorville sekä sorvin oven sulkemis- ja avaamiskäskyt robotilta. Koska laitteiden välillä on jo toimiva kenttäväyläyhteys, tätä päätettiin hyödyntää myös teräkorjaimen siirtämiseen.

Kun teräkorjain on muodostettu robotilla, se pitää siirtää sorvin ohjaukseen. Asia sinänsä on melko suoraviivainen ja alustavasti sen suunniteltiin toimivan näin:

- Tieto korjaimesta siirretään robotilta sorville digitaalisanana Profibus-kenttäväylän yli. Yksi bitti toimii liipaisutietona ja loput bitit ovat varsinaista dataa.
- Sorvin logiikka lukee databitit liipaisubitin nousevalla reunalla ja tallentaa arvon sisäiseen muistipaikkaan.
- Sorvausohjelman alussa sekä työkaluvaihdon yhteydessä suoritetaan sorvilla makro, joka lukee korjaintiedon sisäisestä muistipaikasta ja sijoittaa sen teräkorjaimeen.



KUVA 40. *Robotin I/O-konfigurointivalikko.*

Robotti ja sorvi on liitetty toisiinsa Profibus-kenttäväylän avulla. Liitännästä ei ole varsinaista dokumentaatiota saatavilla, vaan tiedot käytössä olevista biteistä piti kaivaa itse esiin. Robotin puolelta tilanne on selkeä, koska I/O-listasta voidaan helposti katsoa, mitkä bitit on konfiguroitu Profibus-alueelle. Kuvassa 54 on esitetty robotin I/O-konfigurointivalikko. Robotilla on kaksi Profibus-väylään konfiguroitua I/O-aluetta, joista ensimmäinen on varattu kommunikoin-

tiin ylätason ohjauksen kanssa (Fastems MMS) ja jälkimmäinen on konfiguroitu sorville. Sorville on varattu lähdöt 169–208, 40 bittiä eli 5 tavua. Tulotiedot on konfiguroitu alueelle 241–296, 56 bittiä eli 7 tavua.

Sorvin ohjauslogiikkaa on mahdollista seurata ohjausyksikön näytöltä. Logiikka on hyvin laaja, joten sen läpi selaaminen on epärealistisen suuri työ. Ohjelman alusta kuitenkin löytyi kolme riviä, joiden kommentit olivat ”M20 robot call”, ”M20 robot call (M->R)” ja ”robot call finish”. Robotin lähtö 170 on kommentoitu ”M20 kuittaus”, joten sorvin logiikkaruudulta seurattiin näitä kolmea riviä samalla kun robotin lähtöä 170 kytkettiin vuorotellen päälle ja pois. Logiikassa vilkkui kontakti R3000.1 eli tavun 3000 toinen bitti. Koska lähtö 170 on sorville menevän Profibus-alueen toinen bitti, tuntui loogiselta ajatella, että robotin ja sorvin välinen I/O-alue alkaa sorvin muistitavusta 3000.

Tämä ei vielä ratkaissut varsinaista ongelmaa. Oli välttämätöntä saada selville, miten monta bittiä tiedonsiirtoon on varattu ja miten monta niistä on vapaana käyttöä varten – vai onko yksikään.

Ongelmaa yritettiin lähestyä lataamalla logiikkaohjelma tietokoneelle tarkempaa tutustumista varten. Tarkoitus oli hakea logiikkaohjelmasta kaikki robottiin viittaavat kytkennät ja sillä tavoin varmistua, mitkä bitit ovat käytettävissä teräkorjaimen siirtoa varten. Sorvin ohjausyksikössä on 25-napainen sarjaporttiliitin, mutta käyttöoppaissa ei käsitellä tietokoneen logiikkaeditorin ja sorvin välistä yhteyttä. Asiaa päätettiin tiedustella sorvin maahantuojan, Oy Fastems Ab:n asiantuntijoilta. Lyhyt puhelinkeskustelu toi esille kolme tärkeää asiaa:

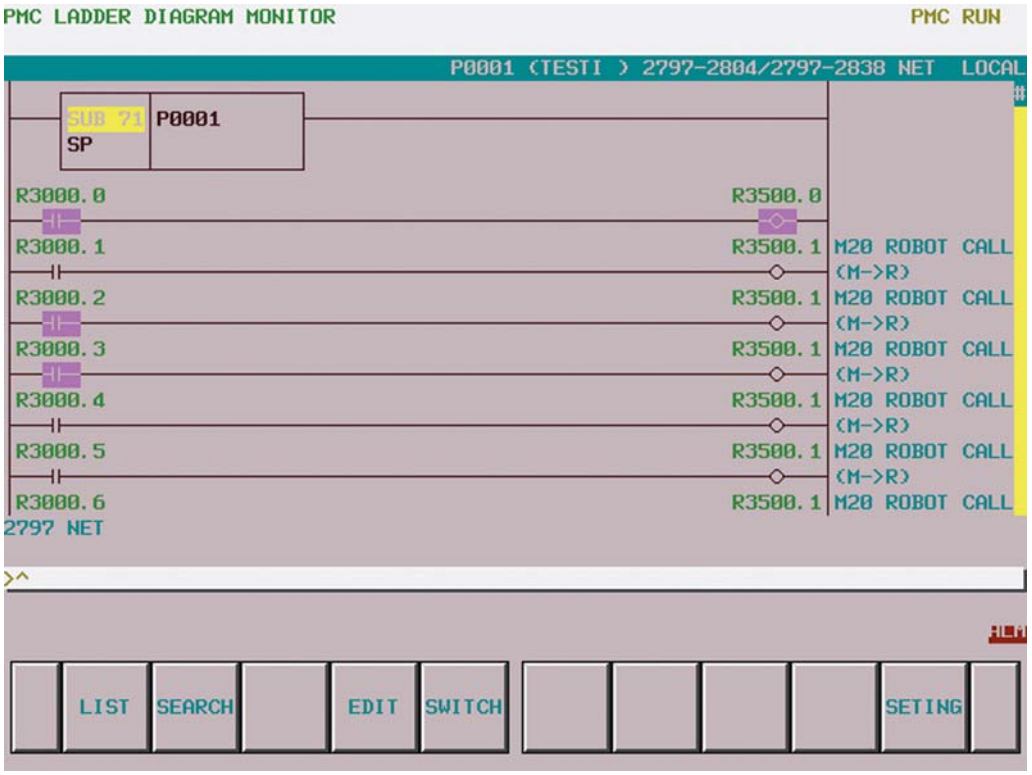
- Profibus-signaalit ovat sorvilla alueilla R3000/R3500.
- Kaikki signaalit, joita ei käytetä logiikkaohjelmassa, ovat vapaasti käytettävissä.
- Sorvin logiikan voi ladata muistikortilla tietokoneelle. (Puhelinkeskustelu, Jari Kuivas, 4.5.2010.)

Puhelinkeskustelu siis vahvisti tiedon siitä, että robotin lähdöt alkavat robotin muistitavusta R3000.

Sorvin ohjauslogiikka saatiin ladattua tietokoneelle muistikortin avulla. Logiikkaohjelmaa tutkittiin Fanuc Ladder III -logiikkaeditorilla. Editoriohjelmasta ei ollut varsinaista ohjekirjaa, mutta yksinkertaisen ohjelman perustoiminnot oppi nopeasti kokeilemalla.

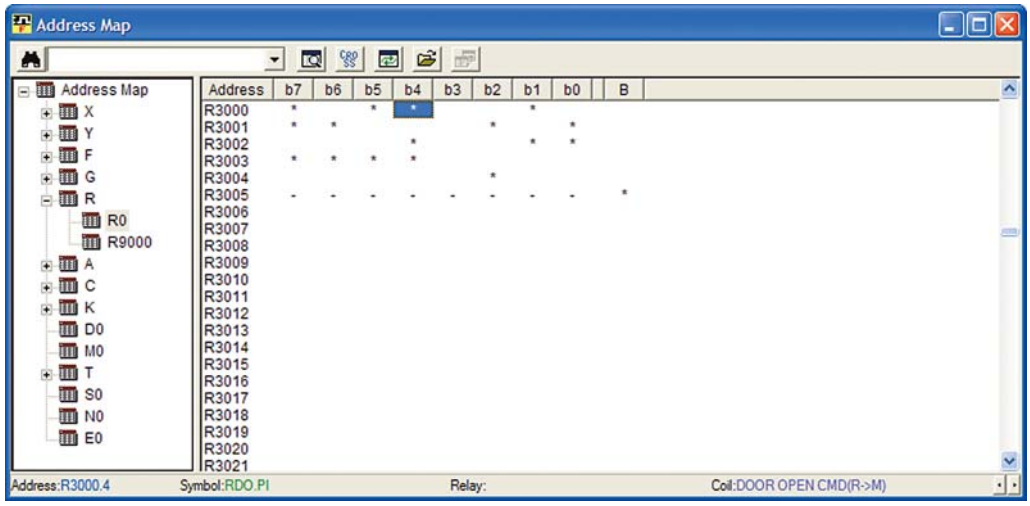
Robotin lähtöjen ja sorvin logiikan muistipaikkojen välinen yhteys päätettiin vielä varmistaa käytännön kokeella. Pohjatietona oli, että robotin lähtö 169 on ensimmäinen lähtö sorvin kanssa yhteisellä Profibus-alueella ja että sorvin päässä Profibus-alue alkaa osoitteesta R3000.0. Tehtiin yksinkertainen muutos logiikkaohjelmaan, missä luetaan bitti kerrallaan viisi tavua alkaen osoitteesta R3000.0 ja päättyen osoitteeseen R3004.7. Logiikkaohjelma ladattiin sorville ja sitä seurattiin sorvin näytöltä. Kun robotin lähtöjä kytkettiin päälle, vastaavat

tulobitit aktivoituivat sorvin logiikkanäkymässä. Kuvassa 25 näkyy, miten osa logiikan kontakteista on aktiivisia. Testin perusteella voitiin varmentaa, millä alueella robotin lähdöt ovat sorvin ohjauksessa. Nämä tiedot kirjoitettiin liitteen 4 taulukkoon.



KUVA 41. *Testiohjelma sorvin ohjauslogiikassa.*

Kun robotin lähtöjen ja sorvin tulojen välinen vastaavuus oli varmennettu, piti vielä selvittää, mitkä bitit ovat vapaina käyttöä varten. Logiikkaeditorissa on address map -näkymä (kuva 56), johon on listattu käytössä olevat muistipaikat. Robotin I/O-listan kommentti ja address mapissa näkyvät logiikkaohjelman I/O-kommentoinnit kirjattiin liitteen 2 taulukkoon. Address mapissa oleva tieto muistipaikkojen käytöstä varmistettiin hakemalla logiikkaeditorin etsi-toiminnolla, käytetäänkö kyseistä muistipaikkaa ohjelmassa. Näin taulukkoon saatiin listattua kaikki todellisuudessa käyttöä varten vapaana olevat bitit. Myöhemmin taulukkoon kirjattiin tiedonsiirtoon tarvittavat bitit, jotka kommentoitiin myös sorvin logiikkaan ja robotin I/O-listaan.



KUVA 42. *Address map.*

15.3 MAKRO-OHJELMA

Sorvin työohjelmissa voidaan käyttää aliohjelmia ja makroja. Makro-ohjelmat on tarkoitettu yleiskäyttöisiksi ohjelmiksi ja niiden merkittävin ero aliohjelmiin on se, että makrolle voidaan antaa ohjelmakutsun yhteydessä parametreja. (Fanuc Ltd. a 2001, 293).

Makro-ohjelma voidaan käynnistää usealla eri tavalla. Yksinkertainen makrokutsu on G-koodi G65. Ohjelmakutsu on muotoa G65 P[ohjelma] L[toistokerrat] [muuttujaparametrit], missä L on valinnainen parametri ja sen oletusarvo on 1. Muuttujaparametrit määritellään kirjaimina, syntaksi on kuvattu taulukossa n. Esimerkiksi ohjelmakutsu G65 P9010 A5.0 käynnistää makro-ohjelman O9010, suorittaa sen yhden kerran ja antaa muuttujalle 1 arvoksi 5.0. (Fanuc Ltd. a 2001, 317).

Tyypillisesti korkean tason ohjelmointikielissä voidaan nimetä muuttujat vapaasti. Esimerkiksi tässä työssä robotin ohjelmointiin käytetyssä Karel-ohjelmointikielessä muuttujille annetaan sisältöä tai käyttötarkoitusta kuvaavat nimet. Sorvin makro-ohjelmoinnissa muuttujat tunnistetaan numeroilla, niitä ei siis nimetä. Muuttujan tunnisteena käytetään merkkiä #. Esimerkiksi muuttujaan 1 viitataan makrossa merkinnällä #1. Muuttujaan 1 voidaan asettaa arvo 3,14 yksinkertaisesti merkinnällä #1=3.14. Desimaalierottimena käytetään siis pistettä. (Fanuc Ltd. a 2001, 294).

15.4 KORJAINARVON KÄSITTELEMINEN ROBOTIN OHJAUKSESSA

Sorvin teräkorjaimen arvo voidaan laskea analysointimoduulin avulla. Teräkorjain lähetetään sorville kolmen eri signaalin avulla:

- liipaisubitillä, joka kertoo robotille, milloin data on luettavissa
- etumerkkibitillä, joka kertoo siirretäänkö negatiivinen vai positiivinen korjain
- varsinaisilla databiteillä, jotka kertovat rinnakkaismuotoisena digitaalisanana korjaimen suuruuden.

Kuten muistakin ohjelmista, myös teräkorjaimen siirtävästä ratkaisusta halutaan tehdä joustava ja helposti konfiguroitava. Tavoitteena on, että loppukäyttäjän ei tarvitse muokata Karel-lähdekoodia. Tähän tavoitteeseen voidaan päästä kahdella tavalla. Ensimmäinen tapa on tehdä Karel-ohjelmasta niin joustava, että sille voidaan syöttää kaikki tarvittava tieto parametreina. Tarvittavia tietoja ovat

- liipaisubitin numero
- etumerkkibitin numero
- databittien alku
- databittien määrä
- skaalaus, joka kertoo mikä on absoluuttimittojen ja digitaaliviestin välinen muuntosuhde (esimerkiksi 5-bittinen siirtokoodi voi saada maksimissaan arvon 2^5-1 eli 31 - tässä skaalaus voisi olla esimerkiksi 100, jolloin digitaalisanan arvo 31 vastaisi teräkorjaimen arvoa 0,31 mm).

Tällaisen Karel-ohjelman toteuttaminen ei ole erityisen hankalaa, mutta käyttäjän kannalta on selkeämpää, jos ohjelmakutsussa säädettäviä parametreja on vähemmän. Parametrien määrää voidaan toki yksinkertaistaa kapseloimalla korjainohjelman kutsu omaan aliohjelmaansa, jolloin käyttäjälle näkyy vain yksinkertainen ohjelmakutsu. Tämän kutsuttavan työnohjelman sisälle puolestaan voidaan sijoittaa Karel-ohjelman kutsu parametreineen, jolloin saavutetaan täysin sama toiminnallisuus ja säädettävyys kuin edellä, mutta varsinaisessa työohjelmassa kompleksisuus on piilotettu ylimääräisen aliohjelman taakse.

Toinen vaihtoehto teräkorjaimen lähettämiseen on toteuttaa koko asia robotin työohjelmana, joka otetaan käyttöön normaalilla aliohjelmakutsulla. Tämä mahdollistaa yhden tärkeän asian: ohjelman rakennetta ja toiminnallisuutta on mahdollista muokata ilman erillisiä ohjelmointityökaluja ja Karel-kokemusta.

Fanucin ohjauksessa on mahdollista niputtaa digitaalilähtöjä ryhmiksi, joista käytetään nimitystä group output (Fanuc Robotics a, 50). Ryhmälähtöä voi tässä tapauksessa käyttää korjaintiedon siirtämiseen helposti. Lisäksi pitää antaa etumerkki ja liipaisutieto.

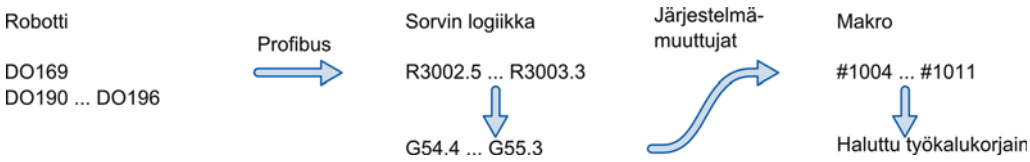
Kolmas vaihtoehto on toteuttaa korjaimen lähetys Karel-ohjelmalla, jossa parametreja ei anneta ohjelmakutsun yhteydessä, vaan ne määritellään suoraan muuttujina, jotka käyttäjä voi konfiguroida ohjelman asennuksen jälkeen. Ohjelman muuttujat ovat luonteeltaan pysyviä. Esimerkiksi liipaisuun käytettävän lähdön numero on jokaisella ohjelmakutsulla sama. Siksi muuttujat voidaan alustaa kerran halutuiksi ja niiden määrittäminen voidaan jättää pois ohjelmakutsusta. Tämä yksinkertaistaa ohjelmakutsua merkittävästi, ja siksi korjaimen lähettävä ohjelma toteutettiinkin tällä menettelyllä. Ohjelmakoodi on liitteessä 10.

Karel-ohjelman muuttujien muuttaminen on käsitelty liitteen 12 osiossa asetusten muuttaminen.

15.5 KORJAINARVON KÄSITTELEMINEN SORVIN OHJAUKSESSA

Korjainarvo kulkee robotilta Profibus-väylän kautta sorville, missä logiikkaohjelma laittaa databitit muistiin. Tieto tallennetaan logiikan ja makrojen väliseen kommunikointiin varatulle alueelle, josta ne voidaan myöhemmin lukea makro-ohjelmalla. Sorvauksen työkierron alussa suoritettava makro katsoo, onko muistipaikoissa korjaindataa ja tarvittaessa lisää uuden korjaimen vanhaan työkalukorjaimeen. Tiedon kulku robotilta sorville on esitetty kuvassa 57.

Robotilta tuleva korjaintieto kierrätetään sorvin logiikan kautta makrolle. Logiikan ja makron välistä tiedonsiirtoa varten ovat käytössä muistipaikat #1000 - #1015, joiden avulla voidaan siirtää tietoa bitti kerrallaan logiikasta makrolle (Fanuc Ltd. 2001, 298).



KUVA 43. *Kommunikointi robotin ohjauksen, sorvin logiikan ja sorvin makron välillä.*

Logiikan ja makron väliset muistipaikat ovat osittain muussa käytössä. Kuten robotin ja sorvin välinen tiedonsiirto, tämäkin varmennettiin etsimällä muistipaikkoja logiikkaohjelmasta. Liitteessä 3 olevaan taulukkoon on merkitty logiikan käyttämät signaalit. Vapaina olevia signaaleja voidaan käyttää tiedonsiirtoon. Muistipaikat, joita teräkorjaimen siirtoon käytetään, on kommentoitu sorvin logiikkaan sekä liitteessä 3 olevaan taulukkoon.

15.5.1 Logiikkaohjelma

Sorvin logiikkaohjelmaan tehtiin Fanuc Ladder III -logiikkaeditorilla aliohjelmakorjain. Sen toiminta on varsin yksinkertainen: robotilta tulevan liipaisubitin nousevalla reunalla luetaan databitit muistiin ja siirretään ne sorvin muistialueelle. Muistialue tyhjennetään sorvin makrolta tulevan nollausbitin nousevalla reunalla.

Koska databitit alkavat tavun R3000 keskeltä, pitää data siirtää bitti kerrallaan. Tässäkin toiminta on hyvin yksinkertainen: nousevan reunan tunnistus asettaa väliaikaisen muistibitin, jonka perusteella sorvin databittien tila asetetaan. Jokaisen databitin kohdalla käydään kaksi lausetta läpi: jos robotilta tuleva signaali on alatilassa ja väliaikainen nousevan reunan muistibitti ylätilassa, resetoi sorvin databitti. Jos sekä robotin signaali että nousevan reunan muistibitti ovat ylätilassa, aseta sorvin databitti ylätilaan. Logiikkaohjelma on liitteessä 4.

Logiikkaohjelma siirtää databitit aina samaan muistipaikkaan ja vasta makro-ohjelma päättää, mihin työkalukorjaimeen arvot lisätään. Nykyisellä ratkaisulla ei siis ole mahdollista siirtää montaa työkalukorjainta yhdellä kertaa. Tähän voidaan jatkokehityksessä soveltaa kahta eri ratkaisua.

Ensimmäinen vaihtoehto on muuttaa laitteiden välistä tiedonsiirtoa ja logiikkaohjelmaa. Robotin ja sorvin väliseen tiedonsiirtoon voidaan tehdä yksinkertainen muutos. Nyt siirretään kolmen tyyppistä dataa: liipaisu, varsinainen data ja etumerkki. Databiteistä voidaan varata yksi kertomaan, siirretäänkö työkalun numero vai korjainarvo. Tämä supistaa korjainarvon laajuuden puoleen nykyisestä, mutta jos väylässä on vapaata kapasiteettia, voidaan databittejä ottaa tarvittaessa lisää käyttöön.

Sorvin logiikka tarkistaa liipaisubitin nousevalla reunalla, onko tyyppibitin perusteella tulossa työkalun numero vai korjaimen arvo. Työkalun numero tallennetaan haluttuun muistipaikkaan. Työkalukorjaimen arvo lähetetään työkalun numeron jälkeen ja sorvin logiikka tallentaa arvon muistipaikkaan, jonka osoite määräytyy työkalun numeron perusteella. Tämä edellyttää joko epäsuoraa muistiviittausta tai hyvin pitkää kiinteästi ohjelmoitua logiikkaohjelmaa. Tämän työn laajuudessa ei selvitetty, onko Fanucin logiikassa mahdollista tehdä epäsuoria muistiviittauksia.

Toinen vaihtoehto on keskustella aktiivisemmin robotin ja sorvin välillä. Nykyisessä ratkaisussa tietoa siirretään vain robotilta sorville. Toteutus voitaisiin tehdä myös niin, että sorvin makro-ohjelma pyytää robotilta työkalun korjaimen ja kertoo samalla Profibus-väylän kautta työkalun numeron. Robotin ohjelma voidaan konfiguroida käynnistymään tulosignaalista, jolloin sorvin makro itse asiassa käynnistää robottiohjelman, joka lukee Profibus-alueelta digitaalisanan ja palauttaa sen perusteella oikean korjaimen. Tämän toteuttaminen ei todennäköisesti ole erityisen vaikeaa.

15.5.2 Makron toteutus

Työkalukorjaimen arvo siirretään sorvin ohjaukseen makro-ohjelman avulla. Tieto työkalukorjaimen arvosta kulkee sorvin ohjauslogiikan kautta. Muistibitit #1004-#1010 on varattu korjaimen arvoa varten ja muistibitti #1011 kertoo etumerkin.

Ohjelmakutsun yhteydessä annettava muuttuja #1 pitää sisällään tiedon työkalun numerosta. Työkalujen X-suuntaiset korjaimet alkavat muistipaikasta #2001, mikä tarkoittaa ensimmäistä työkalukorjainta. Viimeinen työkalukorjain on vastaavasti #2064. Muuttujien avulla voidaan siis siirtää tietoa 64 eri työkalukorjaimen muistipaikkaan. (Fanuc Ltd. 2001, 298.)

Ohjelmassa on lisäksi tarkistus virheelliselle työkalukorjaimen numerolle. Makro hyväksyy vain arvot välillä 1–64.

Koska tieto logiikalta makrolle siirretään bitti kerrallaan, pitää databiteistä koota desimaaliluku makron laskentatoiminnoilla. Bitit eivät ala heti tavun alaosasta, koska neljä ensimmäistä bittiä on varattu logiikan ja makrojen muuhun kommunikointiin. Muunnos databiteistä desimaaliluvuksi päätettiin tehdä peräkkäisillä kahden potenssin kertolaskuilla, missä aina edelliseen arvoon lisätään kunkin databitin kahden potenssi, jos databitti on ylhäällä.

Korjainta pitää voida muuttaa kahteen suuntaan. Siksi on varattu yksi databiteistä kuvaamaan siirrettävän korjaimen etumerkkiä. Etumerkki kulkee muistibitissä #1011. Kun databitit on muutettu desimaaliluvuksi, tarkistetaan etumerkkibitin tila ja jos se on ylhäällä, muutetaan korjain negatiiviseksi kertomalla se -1:llä.

Kuudella databitillä on mahdollista siirtää maksimissaan 26 eli 64 eri arvoa. Vaikka nollan suuruista teräkorjaimen muutosta ei tarvitse lähettää käytännössä milloinkaan, on tiedonsiirron ja toimintalogiikan yksinkertaistamiseksi valittu niin, että korjain voi saada arvoja välillä 0–63. Teräkorjaimen suurin mahdollinen muutos voidaan rajoittaa melko pieneksi, koska korjaimella kompensoidaan nimenomaan teräpalojen kulumista, joka on kappaleiden välillä hyvin pientä. Pienimmäksi inkrementiksi valittiin 0,001 mm, jolloin työkalukorjaimen arvoalueeksi saadaan -0,063–0,063 mm.

Kun makro on lisännyt robotilta saadun työkalukorjaimen arvon vanhaan arvoon, annetaan logiikalle nollaustieto muistibitin #1104 avulla. Logiikka seuraa tämän bitin tilaa ja nollaa nousevalla reunalla työkalukorjaimen siirtoon käytetyt databitit. Tällä vältytään siltä, että teräkorjainta muutetaan liian paljon, mikäli makro suoritetaan monta kertaa peräkkäin.

15.6 RAJOITUKSET JA JATKOKEHITYS

Vaikka kokonaisuus täyttääkin asetetut vaatimukset, se ei suinkaan ole täydellinen. Ohjelmakokonaisuuteen liittyy joitakin teknisiä rajoituksia ja puutteita. Lisäksi on monia asioita, joita voisi kehittää eteenpäin.

15.6.1 Tunnetut rajoitukset

Sorvin teräkorjain on suunniteltu toimimaan vain yhdellä työkalulla. Nykyisessä versiossa on mahdollista mitata kappaleesta eri terillä sorvattuja mittoja ja analysoida kaikki mitat, mutta vain yhden terän korjainarvo pystytään siirtämään ilman erillisiä järjestelyjä.

Kun teräkorjain lähetetään sorville, se kirjoitetaan muistiin edellisen korjaimen päälle. Normaalikäytössä edellinen korjain on jo kirjoitettu työkaluparametreihin makrolla, mutta on silti tärkeää tiedostaa, että robotilta lähetettävä korjain korvaa logiikan väliaikaisessa muistissa olevan korjaimen nykyisessä toteutuksessa.

Mittalaitteiden rajapinnat eivät tunnista virhettä, jos mittalaitteen tiedonsiirtokaapeli on viallinen tai kytketty irti. Keyencen rajapinnassa rajoitus koskee myös sitä tilannetta, että mittalaite on pois päältä. Tähän ei työn tekemisen aikana löydetty toimivaa ratkaisua. Rajoitus pitää ottaa käytössä huomioon. Mittalaitetta käyttävä rajapintaohjelma jää jumiin, ellei saa yhteyttä mittalaitteeseen.

Tietojen kirjaamista koskevat seuraavat rajoitukset:

- Yhteen tiedostoon kirjattavien tietokenttien määrä on rajallinen. Tämä johtuu robotin ohjausjärjestelmän rajoituksesta. Yhdelle ohjelmakutsulle on mahdollista antaa korkeintaan kymmenen parametria (Fanuc robotics a, 1075). Päivämääräkenttä ei sisälly tähän määrään, eli tiedostoon voidaan kirjata maksimissaan kahdeksan tietoa päivämäärän ja kellonajan lisäksi. Tiedoston nimi ja tallennuspaikka vievät kaksi parametria.
- Tiedostoon kirjattavan merkkijonon pituus on rajoitettu. Tämä johtuu robotin ohjausjärjestelmän rajoituksesta. Tekstimuotoisen argumentin maksimipituus on 16 merkkiä (Fanuc robotics a, 232).
- Analysointimoduuli ei nykyisellään käsittele yli sadan kappaleen otoksia. Tiedostossa voi olla enemmän mittoja, mutta vain sataa viimeistä voidaan analysoida.

15.7 AJATUKSIA JATKOKEHITYKSESTÄ

Koska robotin ohjausjärjestelmässä on valmiina sulautettu http-palvelin, saattaisi olla mielekästä toteuttaa Karel-ohjelma, jonka avulla mittauspöytäkirjoja voisi selailla ja ladata selainohjelman avulla samaan verkkoon liitetyltä tietokoneelta käsin.

FTP-palvelimelle tallentamista ei selvitetty tässä työssä. Tietoja kirjaavan ohjelman laajentaminen tukemaan FTP-tallennusta saattaisi olla hyödyllinen monessakin tapauksessa. Näin kaikilta tehtaan robottisoluilta saataisiin keskitetysti kerättyä tietoa suoraan palvelintietokoneelle jatkokäyttöä varten.

Mitutoyo-sisämittalaitteen toteutuva näytteenottotaajuus sarjaväylän kautta käytettynä on melko pieni, noin 16 näytettä sekunnissa. Mittauksen nopeus-/tarkkuussuhdetta voi kehittää eri tavoin. Joskus saattaa olla tarpeen tarkistaa, että mittausliike aloitetaan oikeasta kohdasta: oikein tehdyssä mittauksessa minimimitta tulee kaukana ääripisteistä, jossakin mittausliikkeen keskivaiheilla. Jos mittausliike pidetään hyvin pienenä, on tärkeää varmistua, että minimimitta tulee jossain muualla kuin liikkeen ääripäässä. Jos minimiarvo tulee ensimmäisellä tai viimeisellä mittauksella, voidaan olla melko varmoja, että minimiarvo ei ole kappaleen todellinen halkaisija, vaan mittausliike on tehty väärin.

Tietoja kirjaava ohjelma merkitsee päivämäärän robottiohjauksen oletusmuodossa. Se ei välttämättä ole optimaalinen jatkokäyttöä ajatellen, joten ohjelmaa voisi kehittää antamaan päivämäärän helpommin käsiteltävässä muodossa.

LÄHTEET

Tekes, 2010, SISU 2010 – uusi tuotantoajattelu, Helsinki: Libris Oy.

Teknologiateollisuus ry, www.teknologiateollisuus.fi.

Tilastokeskus, 2008, Teollisuuden toimialakatsaus I/2008, www.stat.fi.

Tilastokeskus, 2008, Teollisuuden toimialakatsaus IV/2008, www.stat.fi.

Tilastokeskus, 2009, Teollisuuden toimialakatsaus IV/2009, www.stat.fi.

Tilastokeskus, 2010, Teollisuuden toimialakatsaus I/2010, www.stat.fi.

Projektiraporttien lähteet:

Andersson Paul H., Tikka H. 1997: Mittaus- ja laatutekniikat. WSOY, Porvoo.

Aumala, Olli 1989: Mittaustekniikan perusteet. Otakustantamo.

Keyence 2001: User's Manual, LS-7500 Series. Keyence Corporation, Osaka, Japani.

Lähteenmäki, Ilkka 1990: Mittaustekniikka 1, 6. painos. Turku.

SFS 2001: SFS-käsikirja 19: Suureet ja yksiköt. SI-mittayksikköjärjestelmä 2001, 3. uudistettu painos. Suomen Standardoimisliitto, Helsinki.

Julkaisemattomat lähteet:

Glouchenko Dmitri & Piira Marko, projektiraportti.

Lento Jaakko, Raita Juuso & Unkuri Tommi, projektiraportti.

Merio Antti, projektiraportti.

Merio Antti & Voltti Gaius, projektiraportti.

Opinnäytetyöt:

Koivunen Sakari, 2010, Mittaaminen robotisoidussa valmistusolussa.

Andersson, P. & Tikka, H. 1997. Mittaus- ja laatutekniikat. Porvoo: WSOY.

Ansaharju, T. & Maaranen, K. 2000. Koneistus. 2. painos. Porvoo: WSOY.

Fanuc Ltd. 2001. Fanuc Series 18i-TB Operator's manual B-63524EN/01.

Fanuc Robotics a. Fanuc robot series R-30iA Handling Tool, operator's manual B-8259EN-2/01.

Fanuc Robotics b. Fanuc robot series R-30iA controller maintenance manual B-82595EN-1/01.

Fanuc Robotics. 2006. Fanuc Robotics SYSTEM R-J3iC Controller KAREL Reference Manual.

Ford, J. 2007. Programming for the Absolute Beginner. Course Technology.

Gnome bugzilla 2010. Python plugins do not work in Gedit for Windows. Viitattu 7.6.2010. https://bugzilla.gnome.org > virheraportti 584679

Haltsonen, S. & Rautanen, E. 2008. Tietokonetekniikka. Helsinki: Edita Prima Oy.

Henttonen, J. & Peltomäki, J. & Uusitalo, S. 2004. Tekniikan matematiikka 2. Helsinki: Edita Prima Oy.

Holopainen, M. & Pulkkinen, P. 1999. Tilastolliset menetelmät. 5. painos. Porvoo: WSOY.

Ihalainen, E. 1978. Konepajan mittaukset. Espoo: Otapaino.

Järnefelt, G. 1990. Tuoteprosessien tilastollinen valvonta - SPC. Tampere: Metalliteollisuuden Kustannus Oy.

Kairisto-Mertanen, L. & Kanerva-Lehto, H. & Penttilä, T. (toim.). 2009. Kohti innovaatiopedagogiikkaa – Uusi lähestymistapa ammattikorkeakoulujen opetukseen ja oppimiseen. Turku: Turun ammattikorkeakoulu.

Keinänen, T. & Räsänen, O. 1983. Mittaustekniikka 2. Porvoo: WSOY.

Koneteknologiakeskus Turku Oy 2010. Viitattu 8.6.2010. http://www.koneteknologiakeskus.fi/

Koskinen, J. 2006. Mikrotietokonetekniikka, sulautetut järjestelmät. 2. painos. Keuruu: Otavan Kirjapaino Oy.

Kuivanen, R. (toim). 1999. Robotiikka. Talentum Oyj/Metallitekniikka.

Mitutoyo 2010. ABSOLUTE Digimatic Indicator ID-C112G for Bore Gage. Viitattu 28.5.2010 http://www.mitutoyo.co.jp/support/service/manual/pdf/99MAH005B_ID-C112G1.pdf.

Nachi Fujikoshi Corp. AX Controller Operating Manual, Robot Language.

Realterm 2010. Viitattu 29.5.2010 http://realterm.sourceforge.net/

Reunanen, T. 2010. Sisu 2010 -seminaariesitys 9.3.2010.

Salomäki, R. 1999. Suorituskykyiset prosessit - hyödynnä SPC. Jyväskylä: Metalliteollisuuden Kustannus Oy.

SFS 3700. 1998. Metrologia. Perus- ja yleistermien sanasto. 3. painos. Helsinki: Suomen Standardisoimisliitto SFS ry.

SFS. 2003. Suureet ja mittayksiköt. SI-mittayksikköjärjestelmä. 3. painos. Helsinki: Kyriiri Oy.

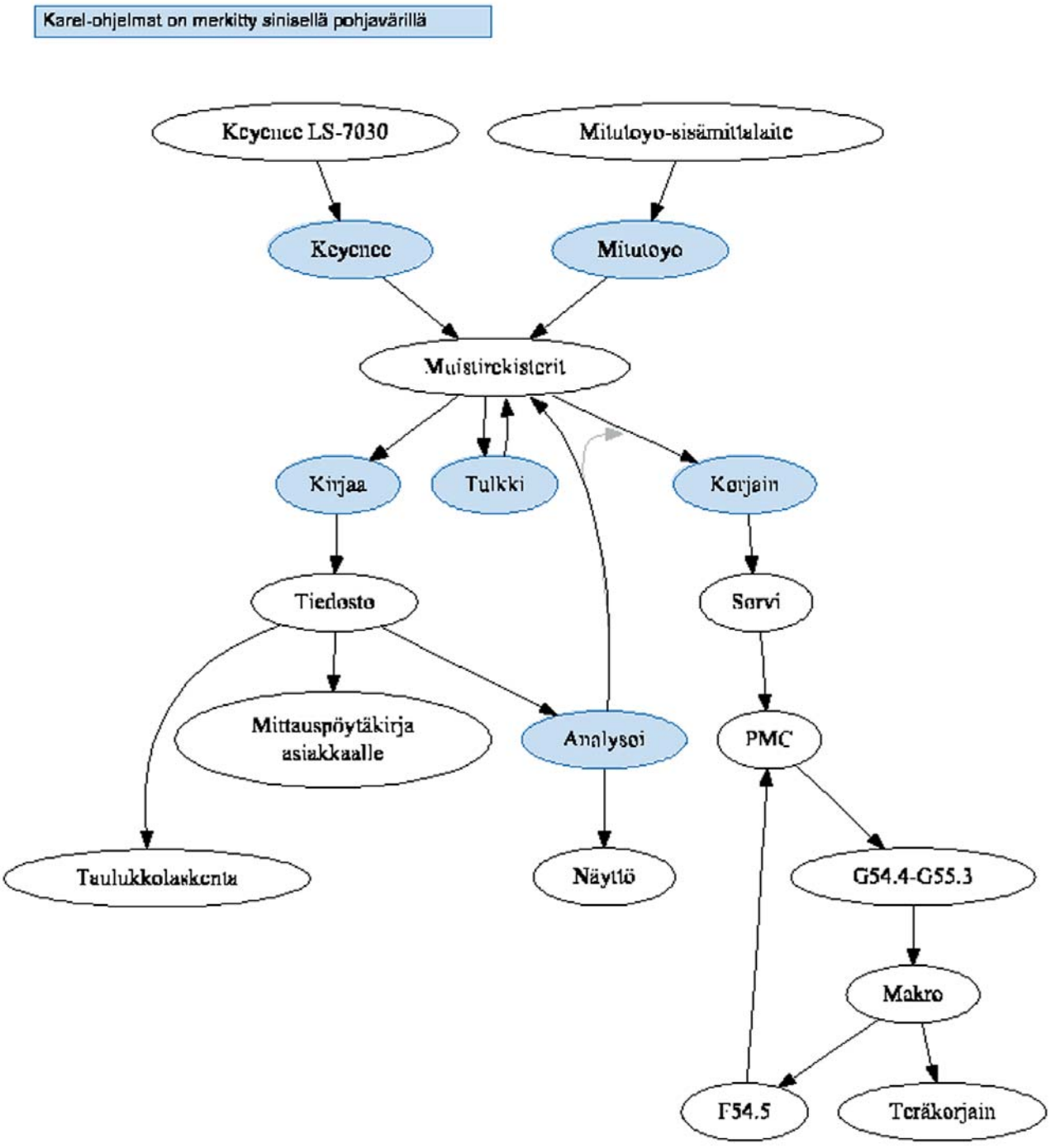
Spiegel M. & Stephens, L. 1999. Statistics. 3. painos. McGraw-Hill.

Turun Ammattikorkeakoulu. Tutkimus- ja kehitustoiminta. Viitattu 1.4.2010 http://turkuamk.fi/ > Tutkimus- ja kehitystoiminta.

Turun Ammattikorkeakoulu. Panoste-projektin esite. 2009. Turku: Turun ammattikorkeakoulu.

LIITE 1.

KOMMUNIKOINTIKAAVIO



LIITE 2.

ROBOTIN JA SORVIN VÄLINEN KOMMUNIKOINTI

Sorvi	**Robotti**	**varattu**	**kommentti ladderin address mapissa**	**symboli**	**robotin IO-kommentti**
R3000.0	DO 169	0	***Liipaisutieto***		
R3000.1	DO 170	1			M20 kuittaus
R3000.2	DO 171	0			
R3000.3	DO 172	0			
R3000.4	DO 173	1	Door open cmd (R->M)	RDO.PI	Aukaise ovi
R3000.5	DO 174	1	Door close cmd (R->M)	RDC.PI	Sulje ovi
R3000.6	DO 175	0			
R3000.7	DO 176	1	Program no search (R->M)	RPNS.PI	
R3001.0	DO 177	1	Robot is in the machine (ITK)	RITK.PI	
R3001.1	DO 178	0	Machine cycle start (R->M)	RMDS.PI	
R3001.2	DO 179	1			
R3001.3	DO 180	0			
R3001.4	DO 181	0			
R3001.5	DO 182	0			
R3001.6	DO 183	1	Left chuck unclamp cmd (R->M)	MCU.PI	Avaa vasen pakka
R3001.7	DO 184	1	Left chuck clamp cmd (R->M)	MCL.PI	Sulje vasen pakka
R3002.0	DO 185	1	Left spdl.orient.1cmd (R->M)	MORI1.PI	Orientointi vasen
R3002.1	DO 186	1	Left spdl.orient.2cmd (R->M)	MORI2.PI	Orientointi vasen
R3002.2	DO 187	0			
R3002.3	DO 188	0			
R3002.4	DO 189	1	Left chuck air blow (R->M)		
R3002.5	DO 190	0	***data***		
R3002.6	DO 191	0	***data***		
R3002.7	DO 192	0	***data***		
R3003.0	DO 193	0	***data***		
R3003.1	DO 194	0	***data***		
R3003.2	DO 195	0	***data***		
R3003.3	DO 196	0	***etumerkki***		
R3003.4	DO 197	1	Right chuck unclamp cmd (R->M)	SCU.PI	Avaa oikea pakka
R3003.5	DO 198	1	Right chuck clamp cmd (R->M)	SCL.PI	Sulje oikea pakka
R3003.6	DO 199	1	Right spdl.orient.1cmd (R->M)	SORI.PI	Orientointi oikea
R3003.7	DO 200	1	Right spdl.orient.1cmd (R->M)	SORI2.PI	Orientointi oikea
R3004.0	DO 201	0			
R3004.1	DO 202	0			
R3004.2	DO 203	1	Right chuck airblow (R->M)	RAB.PI	Puhallus oikea
R3004.3	DO 204	0			
R3004.4	DO 205	0			
R3004.5	DO 206	0			
R3004.6	DO 207	0			
R3004.7	DO 208	0			

LIITE 3.

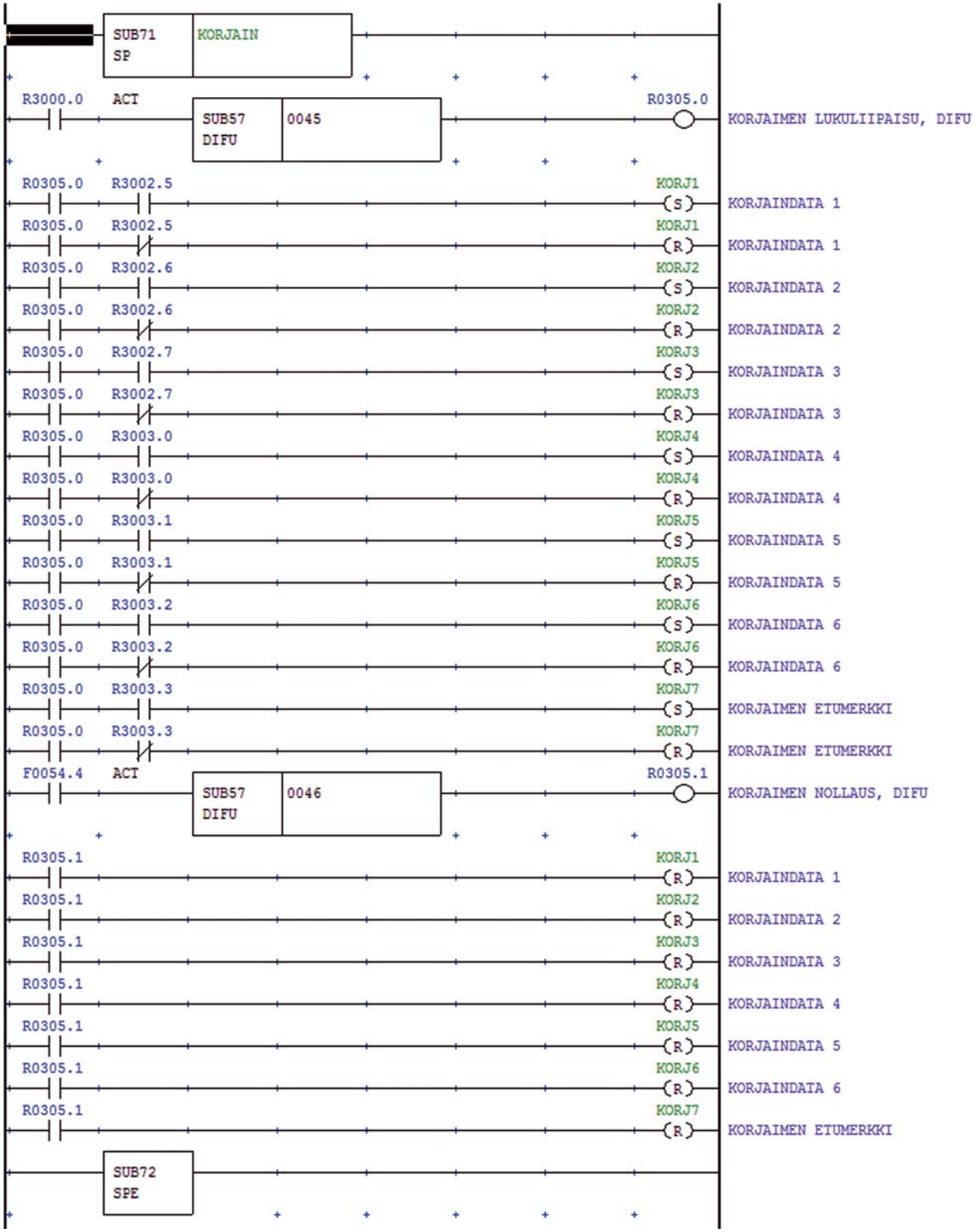
SORVIN LOGIIKAN JA MAKRON VÄLINEN KOMMUNIKOINTI

Logiikalta makrolle		
G54.0	varattu	#1000
G54.1	varattu	#1001
G54.2	varattu	#1002
G54.3	varattu	#1003
G54.4	***liipaisutieto***	#1004
G54.5	***data***	#1005
G54.6	***data***	#1006
G54.7	***data***	#1007
G55.0	***data***	#1008
G55.1	***data***	#1009
G55.2	***data***	#1010
G55.3	***etumerkki***	#1011
G55.4	varattu	#1012
G55.5	varattu	#1013
G55.6	varattu	#1014
G55.7	varattu	#1015

Makrolta logiikalle		
F54.0	varattu	#1100
F54.1	varattu	#1101
F54.2	varattu	#1102
F54.3	varattu	#1103
F54.4	***nollaa***	#1104
F54.5		#1105
F54.6	varattu	#1106
F54.7	varattu	#1107

LIITE 4.

SORVIN LOGIIKKAOHJELMA



LIITE 5.

OHJELMALISTAUS, MITUTOYO

```
PROGRAM mitutoyo


-- Ohjeet kääntäjälle
%COMMENT = 'Mitutoyo'        -- Ohjelman kommentti
%NOLOCKGROUP                 -- Ei liikeryhmiä käytössä
%ENVIRONMENT REGOPE          -- Otetaan rekisterioperaatiot käyttöön
%ENVIRONMENT FLBT


%NOPAUSE = ERROR + COMMAND + TPENABLE
%NOPAUSESHFT
%NOBUSYLAMP


-- Muuttujat
VAR
    sarjaportti :FILE        -- Tiedonsiirtoon käytettävä portti
    vastaus :STRING[30]      -- Mittalaitteelta luettava tulos
    mitta_arvo : REAL        -- Mittatulos liukulukuna
    portti : string[4]       -- Portin tunnus (P2:/P3:)
    rekisteri : INTEGER      --
    kattelyrek : INTEGER     --
    prm_int : INTEGER        --
    prm_real : REAL          --
    real_flag : boolean      --
    prm_str : STRING[8]      --
    datatyyppi : INTEGER     --
    STATUS : INTEGER         --
    minimi : REAL            --
    ajastin : INTEGER        --

BEGIN

    -- Luetaan ohjelmalle annetut parametrit muuttujiin
    -- Portti
    GET_TPE_PRM(1, datatyyppi, prm_int, prm_real, prm_str, STATUS)
    IF STATUS <> 0 THEN
        IF STATUS = 17042 THEN
        WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe! Ohjelma vaatii kolme parametria.')
        ENDIF
        PAUSE
        ABORT
    ENDIF
    IF datatyyppi <> 1 THEN
      WRITE TPERROR (CHR(137) + CHR(128))
        WRITE TPERROR('Parametrin 1 pitää olla kokonaisluku.')
        PAUSE
        ABORT
    ENDIF

    SELECT prm_int OF
    CASE(1):
        portti = 'P2:'
    CASE(2):
        portti = 'P3:'
```

```
  ELSE:
      WRITE TPERROR (CHR(137) + CHR(128))
      WRITE TPERROR('Virhe parametrissa 1! Lue syntaksi ohjeesta.')
      PAUSE
      ABORT
  ENDSELECT

  -- Rekisteri, johon mittatulos sijoitetaan
  GET_TPE_PRM(2, datatyyppi, rekisteri, prm_real, prm_str, STATUS)
  IF STATUS <> 0 THEN
      IF STATUS = 17042 THEN
      WRITE TPERROR (CHR(137) + CHR(128))
          WRITE TPERROR('Virhe! Ohjelma vaatii kolme parametria.')
      ENDIF
      PAUSE
      ABORT
  ENDIF
  IF datatyyppi <> 1 THEN
    WRITE TPERROR (CHR(137) + CHR(128))
      WRITE TPERROR('Parametrin 2 pitää olla kokonaisluku.')
      PAUSE
      ABORT
  ENDIF

  -- Rekisteri, jota käytetään kättelyyn
GET_TPE_PRM(3, datatyyppi, kattelyrek, prm_real, prm_str, STATUS)
  IF STATUS <> 0 THEN
      IF STATUS = 17042 THEN
      WRITE TPERROR (CHR(137) + CHR(128))
          WRITE TPERROR('Virhe! Ohjelma vaatii kolme parametria.')
      ENDIF
      PAUSE
      ABORT
  ENDIF
  IF datatyyppi <> 1 THEN
    WRITE TPERROR (CHR(137) + CHR(128))
      WRITE TPERROR('Parametrin 3 pitää olla kokonaisluku.')
      PAUSE
      ABORT
  ENDIF

  -- Kättelyt
  -- 0: Alku
  -- 1: Robotille tieto -> saa aloittaa liikkeen
  -- 2: Tieto robotilta -> liike tehty
  -- 3: Tieto robotille -> mittatulos luettu muistiin, saa poistua
  -- Nollataan kättelymuuttuja varmuuden vuoksi
  SET_INT_REG(kattelyrek, 0, status)

  -- Avataan sarjaportti
  OPEN FILE sarjaportti('RW',portti)

  -- Luetaan ensimmäinen arvo ja asetetaan se minimiksi
  WRITE sarjaportti ('1', CHR(13))
  READ sarjaportti(vastaus::3::0)
  IF vastaus <> '01A' THEN
      IF vastaus = '911' THEN
          WRITE TPERROR (CHR(137) + CHR(128))
          WRITE TPERROR('Mittalaite pois päältä!')
```

```
        ELSE
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe tiedonsiirrossa! (alku)')
        ENDIF
        CLOSE FILE sarjaportti
        PAUSE
        ABORT
    ENDIF

    READ sarjaportti (vastaus::10::0)
    CNV_STR_REAL(vastaus, minimi)

    -- Annetaan robotille lupa aloittaa mittaus
    SET_INT_REG(kattelyrek, 1, status)

    -- Aloitetaan mittaussilmukka
    REPEAT
        WRITE sarjaportti ('1', CHR(13))
        READ sarjaportti(vastaus::3::0)
        IF vastaus <> '01A' THEN
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe tiedonsiirrossa! (silmukka)')
            CLOSE FILE sarjaportti
            PAUSE
            ABORT
        ENDIF

        READ sarjaportti (vastaus::10::0)
        CNV_STR_REAL(vastaus, mitta_arvo)
        SET_REAL_REG(rekisteri, mitta_arvo, STATUS)
        IF mitta_arvo < minimi THEN
            minimi = mitta_arvo
        ENDIF
        GET_REG(kattelyrek, real_flag, prm_int, prm_real, status)
    UNTIL prm_int = 2

    SET_REAL_REG(rekisteri, minimi, STATUS)
    SET_INT_REG(kattelyrek, 3, status)

    CLOSE FILE sarjaportti

END mitutoyo
```

LIITE 6.

OHJELMALISTAUS, KEYENCE

```
PROGRAM keyence


-- Ohjeet kääntäjälle
%COMMENT = 'Keyence LS-7501'           -- Ohjelman kommentti
%NOLOCKGROUP                          -- Ei liikeryhmiä käytössä
%ENVIRONMENT REGOPE                   -- Otetaan rekisterioperaatiot
käyttöön

%NOPAUSE = ERROR + COMMAND + TPENABLE  -- Conditions for pause
program.
%NOPAUSESHFT                          -- No pause if shift is
released.
%NOBUSYLAMP                           -- Bysylamp and UO[ 3] Prg
running = OFF


-- Muuttujat
VAR
    sarjaportti :FILE
    vastaus :STRING[30]           -- Mittalaitteelta luettava tulos
    mitta_arvo : REAL             -- Mittatulos liukulukuna
    tehtava : INTEGER
    portti : STRING[4]
    parametri : INTEGER
    kattelyrek : INTEGER
    suodatus : STRING[4]
    ohjelmanro : string[2]
    prm_int : INTEGER
    prm_real : REAL
    prm_str : STRING[8]
    datatyyppi : INTEGER
    real_flag : BOOLEAN
    STATUS : INTEGER

ROUTINE mitta : real -- Palauttaa tämänhetkisen mitan
    BEGIN
        -- Asetetaan mittalaite NORMAL-tilaan
        WRITE sarjaportti ('SD,ME,1,00', CHR(13))
        -- Varmistetaan, että tilan vaihto onnistui
        READ sarjaportti(vastaus::7::0)
        IF vastaus <> 'SD,ME,1' THEN
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe tiedonsiirrossa!')
            CLOSE FILE sarjaportti
            PAUSE
            ABORT
        ENDIF

        -- Luetaan CR pois puskurista
        READ sarjaportti(vastaus::1::0)

        -- Luetaan mittatulos
        WRITE sarjaportti ('M1,0', CHR(13))
        -- Luetaan tulosteen alkuosa pois ja käytetään sitä
tarkistamiseen
        READ sarjaportti(vastaus::4::0)
```

```
        IF vastaus <> 'M1,+' THEN
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe tiedonsiirrossa!')
            CLOSE FILE sarjaportti
            PAUSE
            ABORT
        ENDIF

        -- Luetaan varsinainen mittausdata
        READ sarjaportti (vastaus::8::0)
        CNV_STR_REAL(vastaus, mitta_arvo)
        return (mitta_arvo)

    END mitta

ROUTINE alamitta : real -- Palauttaa alimman arvon mittausjaksolta
    -- Kättelyt
    -- 0: Alku
    -- 1: Robotille tieto -> saa aloittaa liikkeen
    -- 2: Tieto robotilta -> liike tehty
    -- 3: Tieto robotille -> mittatulos luettu muistiin, saa poistua
    BEGIN
        -- Nollataan kättelymuuttuja varmuuden vuoksi
        SET_INT_REG(kattelyrek, 0, status)

        -- Asetetaan mittalaite AUTO BOTTOM HOLD -tilaan
        WRITE sarjaportti ('SD,ME,1,06', CHR(13))
        -- Varmistetaan, että tilan vaihto onnistui
        READ sarjaportti(vastaus::7::0)
        IF vastaus <> 'SD,ME,1' THEN
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe tiedonsiirrossa! (asetus)')
            CLOSE FILE sarjaportti
            PAUSE
            ABORT
        ENDIF

        -- Luetaan CR pois puskurista
        READ sarjaportti(vastaus::1::0)

        -- Nollataan vanha mittaustulos
        WRITE sarjaportti ('Q1', CHR(13))
        -- Varmistetaan, että nollaus onnistui
        READ sarjaportti(vastaus::2::0)
        IF vastaus <> 'Q1' THEN
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe tiedonsiirrossa! (nollaus)')
            CLOSE FILE sarjaportti
            PAUSE
            ABORT
        ENDIF

        -- Luetaan CR pois puskurista
        READ sarjaportti(vastaus::1::0)

        -- Aloitetaan mittaus
        WRITE sarjaportti ('U1', CHR(13))
        -- Varmistetaan, että viesti meni perille
        READ sarjaportti(vastaus::2::0)
```

```
        IF vastaus <> 'U1' THEN
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe tiedonsiirrossa! (aloitus)')
            CLOSE FILE sarjaportti
            PAUSE
            ABORT
        ENDIF

        -- Luetaan CR pois puskurista
        READ sarjaportti(vastaus::1::0)

        -- Annetaan robotille lupa aloittaa liike
        SET_INT_REG(kattelyrek, 1, status)

        -- Odotetaan robotin liikkeen loppumista
        REPEAT
            DELAY(100)
            GET_REG(kattelyrek, real_flag, prm_int, prm_real, status)
        UNTIL prm_int = 2

        --  Luetaan mittaustulos
        WRITE sarjaportti ('L1,0', CHR(13))
        -- Varmistetaan, että viesti meni perille
        READ sarjaportti(vastaus::4::0)
        IF vastaus <> 'L1,+' THEN
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe tiedonsiirrossa! (mitta)')
            CLOSE FILE sarjaportti
            PAUSE
            ABORT
        ENDIF
        -- Luetaan varsinainen mittausdata
        READ sarjaportti (vastaus::8::0)
        CNV_STR_REAL(vastaus, mitta_arvo)

        -- Kerrotaan robotille, että mitta on luettu ja voi jatkaa
matkaa
        SET_INT_REG(kattelyrek, 3, status)
        return (mitta_arvo)

    END alamitta


ROUTINE keskiarvo  -- Muuttaa keskiarvoistuksen määrää
    BEGIN
        IF (parametri>=0) and (parametri<=12) THEN
            -- Muutetaan käsky sopivaan muotoon
            CNV_INT_STR(parametri, 0, 0, suodatus)
            -- Leikataan tyhje alusta pois
            suodatus = sub_str(suodatus,2,2)
            IF parametri < 10 THEN
                suodatus = '0' + suodatus
            ENDIF

            -- Lähetetään uusi suodatuksen arvo
            WRITE sarjaportti ('SD,AV,1,', suodatus, CHR(13))
            -- Varmistetaan, että arvon muuttaminen onnistui
            READ sarjaportti(vastaus::7::0)
            IF vastaus <> 'SD,AV,1' THEN
```

```
                WRITE TPERROR (CHR(137) + CHR(128))
                WRITE TPERROR('Virhe tiedonsiirrossa!')
                CLOSE FILE sarjaportti
                PAUSE
                ABORT
            ENDIF
        ELSE
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe parametrissa 3! Lue syntaksi
ohjeesta.')
            PAUSE
            ABORT
        ENDIF

    END keskiarvo


ROUTINE ohjelma  -- Vaihtaa mittausohjelman
    BEGIN
        IF (parametri>=0) and (parametri<16) THEN
            -- Muutetaan käsky heksamuotoon
            CNV_INT_STR(parametri, 0, 16, ohjelmanro)
            -- Leikataan tyhje alusta pois
            ohjelmanro = sub_str(ohjelmanro,2,1)

            -- Lähetetään ohjelman numero mittalaitteelle
            WRITE sarjaportti ('PW,',ohjelmanro, CHR(13))
            -- Kuunnellaan vastaus
            READ sarjaportti(vastaus::2::0)
            IF vastaus <> 'PW' THEN
                WRITE TPERROR (CHR(137) + CHR(128))
                WRITE TPERROR('Virhe tiedonsiirrossa!')
                CLOSE FILE sarjaportti
                PAUSE
                ABORT
            ENDIF
        ELSE
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe parametrissa 3! Lue syntaksi
ohjeesta.')
            PAUSE
            ABORT
        ENDIF
    END ohjelma

BEGIN

    -- Luetaan ohjelmalle annetut parametrit muuttujiin

    -- Portti
    GET_TPE_PRM(1, datatyyppi, prm_int, prm_real, prm_str, STATUS)
    IF STATUS <> 0 THEN
        IF STATUS = 17042 THEN
        WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe! Ohjelma vaatii kolme parametria.')
        ENDIF
        PAUSE
        ABORT
    ENDIF
```

```
IF datatyyppi <> 1 THEN
    WRITE TPERROR (CHR(137) + CHR(128))
    WRITE TPERROR('Parametrin 1 pitää olla kokonaisluku.')
    PAUSE
    ABORT
ENDIF

SELECT prm_int OF
CASE(1):
    portti = 'P2:'
CASE(2):
    portti = 'P3:'
ELSE:
    WRITE TPERROR (CHR(137) + CHR(128))
    WRITE TPERROR('Virhe parametrissa 1! Lue syntaksi ohjeesta.')
    PAUSE
    ABORT
ENDSELECT


-- Tehtävä
-- 1 lue mitta
-- 2 lue alamitta
-- 3 muuta keskiarvoistusta
-- 4 vaihda ohjelmaa
GET_TPE_PRM(2, datatyyppi, tehtava, prm_real, prm_str, STATUS)
IF STATUS <> 0 THEN
    IF STATUS = 17042 THEN
    WRITE TPERROR (CHR(137) + CHR(128))
        WRITE TPERROR('Virhe! Ohjelma vaatii kolme parametria.')
    ENDIF
    PAUSE
    ABORT
ENDIF
IF datatyyppi <> 1 THEN
  WRITE TPERROR (CHR(137) + CHR(128))
    WRITE TPERROR('Parametrin 2 pitää olla kokonaisluku.')
    PAUSE
    ABORT
ENDIF


-- Parametri
-- mittaamisen tulosrekisteri
-- ohjelman vaihtamisessa uuden ohjelman numero (0...F)
-- keskiarvoistuksen muuttamisessa suodatuksen uusi arvo (00...12)
GET_TPE_PRM(3, datatyyppi, parametri, prm_real, prm_str, STATUS)
IF STATUS <> 0 THEN
    IF STATUS = 17042 THEN
    WRITE TPERROR (CHR(137) + CHR(128))
        WRITE TPERROR('Virhe! Ohjelma vaatii kolme parametria.')
    ENDIF
    PAUSE
    ABORT
ENDIF
IF datatyyppi <> 1 THEN
    WRITE TPERROR (CHR(137) + CHR(128))
    WRITE TPERROR('Parametrin 3 pitää olla kokonaisluku.')
    PAUSE
```

```
        ABORT
    ENDIF


    -- Kättelyrekisteri (valinnainen, vain jos tehtävä on alamitan
lukeminen)
    IF tehtava = 2 THEN
        GET_TPE_PRM(4, datatyyppi, kattelyrek, prm_real, prm_str,
STATUS)
        IF STATUS <> 0 THEN
            IF STATUS = 17042 THEN
            WRITE TPERROR (CHR(137) + CHR(128))
                WRITE TPERROR('Virhe! Ohjelma vaatii neljä
parametria.')
            ENDIF
            PAUSE
            ABORT
        ENDIF
        IF datatyyppi <> 1 THEN
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Parametrin 4 pitää olla kokonaisluku.')
            PAUSE
            ABORT
        ENDIF
    ENDIF

-- testataan

    OPEN FILE sarjaportti('RW',portti)

    -- Valitaan oikea aliohjelma tehtäväparametrin perusteella tai
poistutaan, jos parametreissa on epäselvyyttä
    SELECT tehtava OF
    CASE(1):
        SET_REAL_REG(parametri, mitta, STATUS)
    CASE(2):
        SET_REAL_REG(parametri, alamitta, STATUS)
    CASE(3):
        keskiarvo
    CASE(4):
        ohjelma
    ELSE:
        WRITE TPERROR (CHR(137) + CHR(128))
        WRITE TPERROR('Virhe parametrissa 2! Lue syntaksi ohjeesta.')
        PAUSE
        ABORT
    ENDSELECT

    CLOSE FILE sarjaportti

END keyence
```

LIITE 7.

OHJELMALISTAUS, TULKKI

```
PROGRAM tulkki


-- Ohjeet kääntäjälle
%COMMENT = 'Tulkki'
%NOLOCKGROUP
%ENVIRONMENT REGOPE

%NOPAUSE = ERROR + COMMAND + TPENABLE
%NOPAUSESHFT
%NOBUSYLAMP


-- Muuttujat
VAR
    mitta, alaraja, ylaraja, epavarmuus : REAL
    tulosrek : INTEGER
    prm_int : INTEGER
    prm_real : REAL
    prm_str : STRING[8]
    datatyyppi : INTEGER
    real_flag : BOOLEAN
    status : INTEGER

BEGIN

    -- Luetaan ohjelmalle annetut parametrit muuttujiin

    -- Mitta
    GET_TPE_PRM(1, datatyyppi, prm_int, mitta, prm_str, STATUS)
    IF STATUS <> 0 THEN
        IF STATUS = 17042 THEN
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe! Ohjelma vaatii neljä parametria.')
        ENDIF
        PAUSE
        ABORT
    ENDIF
    IF datatyyppi <> 2 THEN  -- Jos parametri ei ole liukuluku
        -- Jos parametri on kokonaisluku, sen arvo tallennetaan,
muuten palautetaan virhe
        IF datatyyppi = 1 THEN
            mitta = prm_int
        ELSE
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Parametrin 1 pitää olla luku.')
            PAUSE
            ABORT
        ENDIF
    ENDIF

    -- Alaraja
    GET_TPE_PRM(2, datatyyppi, prm_int, alaraja, prm_str, STATUS)
    IF STATUS <> 0 THEN
        IF STATUS = 17042 THEN
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe! Ohjelma vaatii neljä parametria.')
```

```
        ENDIF
        PAUSE
        ABORT
    ENDIF
    IF datatyyppi <> 2 THEN  -- Jos parametri ei ole liukuluku
        -- Jos parametri on kokonaisluku, sen arvo tallennetaan,
muuten palautetaan virhe
        IF datatyyppi = 1 THEN
            alaraja = prm_int
        ELSE
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Parametrin 2 pitää olla luku.')
            PAUSE
            ABORT
        ENDIF
    ENDIF

    -- Yläraja
    GET_TPE_PRM(3, datatyyppi, prm_int, ylaraja, prm_str, STATUS)
    IF STATUS <> 0 THEN
        IF STATUS = 17042 THEN
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe! Ohjelma vaatii neljä parametria.')
        ENDIF
        PAUSE
        ABORT
    ENDIF
    IF datatyyppi <> 2 THEN  -- Jos parametri ei ole liukuluku
        -- Jos parametri on kokonaisluku, sen arvo tallennetaan,
muuten palautetaan virhe
        IF datatyyppi = 1 THEN
            ylaraja = prm_int
        ELSE
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Parametrin 3 pitää olla luku.')
            PAUSE
            ABORT
        ENDIF
    ENDIF

    -- Tulosrekisteri
    GET_TPE_PRM(4, datatyyppi, tulosrek, prm_real, prm_str, STATUS)
    IF STATUS <> 0 THEN
        IF STATUS = 17042 THEN
        WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe! Ohjelma vaatii neljä parametria.')
        ENDIF
        PAUSE
        ABORT
    ENDIF
    IF datatyyppi <> 1 THEN
        WRITE TPERROR (CHR(137) + CHR(128))
        WRITE TPERROR('Parametrin 4 pitää olla kokonaisluku.')
        PAUSE
        ABORT
    ENDIF

    -- Mittalaitteen epävarmuus
    GET_TPE_PRM(5, datatyyppi, prm_int, epavarmuus, prm_str, STATUS)
```

```
    IF STATUS <> 0 THEN
        IF STATUS = 17042 THEN
            epavarmuus = 0
            GO TO loppu
        ENDIF
        WRITE TPERROR (CHR(137) + CHR(128))
        WRITE TPERROR('Virhe parametrissa 5!')
        PAUSE
        ABORT
    ENDIF
    IF datatyyppi <> 2 THEN  -- Jos parametri ei ole liukuluku
        -- Jos parametri on kokonaisluku, sen arvo tallennetaan,
muuten palautetaan virhe
        IF datatyyppi = 1 THEN
            epavarmuus = prm_int
        ELSE
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Parametrin 5 pitää olla luku.')
            PAUSE
            ABORT
        ENDIF
    ENDIF

    loppu::

    -- Jos ollaan alatoleranssin alapuolella, palautetaan -2
    IF (mitta-epavarmuus)<alaraja THEN
        SET_INT_REG(tulosrek, -2, status)
    ELSE
    -- Jos ollaan ylätoleranssin yläpuolella, palautetaan -1
        IF(mitta+epavarmuus)>ylaraja THEN
            SET_INT_REG(tulosrek, -1, status)
        ELSE
            -- Ei ali eikä yli -> OK ja palautetaan 1
            SET_INT_REG(tulosrek, 1, status)
        ENDIF
    ENDIF

END tulkki
```

LIITE 8.

OHJELMALISTAUS, KIRJAA

```
PROGRAM kirjaa

%COMMENT = 'Kirjaa tiedostoon'
%ENVIRONMENT TIM
%ENVIRONMENT STRNG
%ENVIRONMENT FDEV

VAR
    lokitiedosto : FILE
    testitied : FILE
    aika_int : INTEGER
    aika_str : STRING[30]
    kirjaus : STRING[200]
    aika: INTEGER
    muistilaite : string[8]
    tiedosto : STRING[30]
    lisatieto: STRING[16]
    i : INTEGER
    prm_int : INTEGER
    prm_real : REAL
    prm_str : STRING[8]
    STATUS : INTEGER
    datatyyppi : INTEGER

ROUTINE kohde_valmis : boolean
VAR
    testi : string[32]
BEGIN
    write tpdisplay('testi',cr)

    testi = muistilaite + 'testitiedosto.tst'
    open file testitied('RW', testi)
    write testitied('OK..',cr)
    close file testitied
    open file testitied('RO', testi)
    read testitied(testi::2::0)
    IF UNINIT(testi)=true THEN
        close file testitied
        return(FALSE)
    ELSE
        close file testitied
        return(TRUE)
    ENDIF

END kohde_valmis

BEGIN
    kirjaus = ''
    --    1. parametri on tiedostonnimi, pitää olla string-tyyppiä
    GET_TPE_PRM(1, datatyyppi, prm_int, prm_real, tiedosto, STATUS)
    IF ((STATUS <> 0) OR (datatyyppi <> 3)) THEN
        WRITE TPERROR (CHR(137) + CHR(128))
        WRITE TPERROR('Virhe parametrissa 1! Lue syntaksi ohjeesta.')
        PAUSE
        ABORT
    ENDIF
```

```
    --    2. parametri on käytettävä muistilaite
    --    1=CF-muistikortti
    --    2=USB-muisti
    GET_TPE_PRM(2, datatyyppi, prm_int, prm_real, prm_str, STATUS)
    IF ((STATUS <> 0) OR (datatyyppi <> 1)) THEN
        WRITE TPERROR (CHR(137) + CHR(128))
        WRITE TPERROR('Virhe parametrissa 2! Lue syntaksi ohjeesta.')
        PAUSE
        ABORT
    ENDIF
    SELECT prm_int OF
        CASE(1):
            muistilaite = 'MC:'
        CASE(2):
            muistilaite = 'UD1:'
        ELSE:
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe parametrissa 2! Lue syntaksi
ohjeesta.')
            PAUSE
            ABORT
    ENDSELECT

    --    parametrit 3 - 10 ovat valinnaisia tallennettavia tietoja
    i = 3
    WHILE i < 11 DO
          GET_TPE_PRM(i, datatyyppi, prm_int, prm_real, prm_str,
STATUS)
          IF STATUS = 17042  THEN
              datatyyppi = 4
          ENDIF
          SELECT datatyyppi OF
            CASE(1): -- integer
                CNV_INT_STR(prm_int, 1, 0, lisatieto)
                kirjaus = kirjaus + lisatieto + ','
            CASE(2): -- real
                CNV_REAL_STR(prm_real, 1, 8, lisatieto)
                kirjaus = kirjaus + lisatieto + ','
            CASE(3): -- string
                kirjaus = kirjaus + prm_str + ','
            CASE(4): --parametria ei ole, lopetetaan
                i = 99
            ENDSELECT
          i = i +1
    ENDWHILE

    -- Kirjoitetaan viimeiseksi päivämäärä ja kellonaika

    GET_TIME(aika_int)
    CNV_TIME_STR(aika_int, aika_str)
    kirjaus = kirjaus + aika_str

    -- Avataan tiedosto

    tiedosto = muistilaite + tiedosto

    OPEN FILE lokitiedosto('AP', tiedosto)
    IF (IO_STATUS(lokitiedosto) <> 0) or (kohde_valmis=FALSE) THEN
        WRITE TPERROR (CHR(137) + CHR(128))
```

```
        WRITE TPERROR('Virhe tiedoston avaamisessa!')
        PAUSE
        ABORT
    ENDIF
    WRITE lokitiedosto (kirjaus, CHR(13))
    IF (IO_STATUS(lokitiedosto) <> 0) THEN
        WRITE TPDISPLAY('Virhe kirjoitettaessa tiedostoon!', cr)
        PAUSE
        ABORT
    ENDIF
    CLOSE FILE lokitiedosto

END kirjaa
```

LIITE 9.

OHJELMALISTAUS, ANALYSOI

```
PROGRAM analysoi

%COMMENT = 'Analysoi tuloksia'
%ENVIRONMENT STRNG
%ENVIRONMENT FDEV
%ENVIRONMENT REGOPE

CONST
    pituus = 100
VAR
    lokitiedosto, testitied : FILE
    rivi : STRING[50]
    tiedosto : STRING[30]
    prm_int : INTEGER
    prm_real : REAL
    prm_str : STRING[8]
    STATUS : INTEGER
    datatyyppi : INTEGER
    taulukko : ARRAY[pituus] OF REAL
    mitta : REAL
    valiarvo : REAL
    alkiot_lkm : INTEGER
    tehtava : INTEGER
    maara : INTEGER
    merkki : STRING[1]
    i : INTEGER
    tulosrek : INTEGER
    muistilaite : string[8]


ROUTINE kohde_valmis : boolean
VAR
    testi : string[32]
BEGIN
    write tpdisplay('testi',cr)

    testi = muistilaite + 'testitiedosto.tst'
    open file testitied('RW', testi)
    write testitied('OK..',cr)
    close file testitied
    open file testitied('RO', testi)
    read testitied(testi::2::0)
    IF UNINIT(testi)=true THEN
        close file testitied
        return(FALSE)
    ELSE
        close file testitied
        return(TRUE)
    ENDIF

END kohde_valmis


ROUTINE arvo : REAL
BEGIN
    IF maara > alkiot_lkm THEN
        maara = alkiot_lkm
```

```
        WRITE TPERROR (CHR(137) + CHR(128))
        WRITE TPERROR('Palautettiin ', maara, '. arvo.')
    ENDIF
    RETURN (taulukko[alkiot_lkm-(maara-1)])
END arvo

ROUTINE keskiarvo : REAL
BEGIN
    IF maara > alkiot_lkm THEN
        maara = alkiot_lkm
        WRITE TPERROR (CHR(137) + CHR(128))
        WRITE TPERROR('Palautettiin ', maara, ' kappaleen keskiarvo.')
    ENDIF
    valiarvo = 0
    FOR i = 0 to maara-1 DO
        valiarvo = valiarvo + taulukko[alkiot_lkm-i]
    ENDFOR
    RETURN (valiarvo/maara)
END keskiarvo

ROUTINE minimi : REAL
BEGIN
    IF maara > alkiot_lkm THEN
        maara = alkiot_lkm
        WRITE TPERROR (CHR(137) + CHR(128))
        WRITE TPERROR('Palautettiin ', maara, ' kappaleen minimi.')
    ENDIF
    valiarvo = taulukko[alkiot_lkm]
    FOR i = 1 to maara-1 DO
        IF taulukko[alkiot_lkm-i] < valiarvo THEN
            valiarvo = taulukko[alkiot_lkm-i]
        ENDIF
    ENDFOR
    RETURN (valiarvo)
END minimi

ROUTINE maksimi : REAL
BEGIN
    IF maara > alkiot_lkm THEN
        maara = alkiot_lkm
        WRITE TPERROR (CHR(137) + CHR(128))
        WRITE TPERROR('Palautettiin ', maara, ' kappaleen maksimi.')
    ENDIF
    valiarvo = taulukko[alkiot_lkm]
    FOR i = 1 to maara-1 DO
        IF taulukko[alkiot_lkm-i] > valiarvo THEN
            valiarvo = taulukko[alkiot_lkm-i]
        ENDIF
    ENDFOR
    RETURN (valiarvo)
END maksimi

ROUTINE hajonta : REAL
VAR
    ka : REAL
BEGIN
    IF maara > alkiot_lkm THEN
        maara = alkiot_lkm
        WRITE TPERROR (CHR(137) + CHR(128))
```

```
        WRITE TPERROR('Palautettiin ', maara, ' kappaleen
keskihajonta.')
    ENDIF
    ka = keskiarvo
    valiarvo = 0
    FOR i = 1 to maara DO
        valiarvo = valiarvo + (taulukko[alkiot_lkm-(i-1)]-
ka)*(taulukko[alkiot_lkm-(i-1)]-ka)
    ENDFOR
    return (sqrt(valiarvo/maara))
END hajonta


ROUTINE yhteenveto
BEGIN
    IF maara > alkiot_lkm THEN
        maara = alkiot_lkm
        WRITE TPERROR (CHR(137) + CHR(128))
        WRITE TPERROR('Palautettiin ', maara, ' kappaleen
yhteenveto.')
    ENDIF
    WRITE TPDISPLAY(cr,cr,cr,cr,cr,cr,cr)
    WRITE TPDISPLAY('Mittaustulosten analysointi, n =',maara,cr)
    WRITE TPDISPLAY('Aritmeettinen keskiarvo: ',keskiarvo::0::6, cr)
    WRITE TPDISPLAY('Keskihajonta: ',hajonta::0::6, cr)
    WRITE TPDISPLAY('Minimiarvo: ',minimi::0::6, cr)
    WRITE TPDISPLAY('Maksimiarvo: ',maksimi::0::6, cr)
END yhteenveto

ROUTINE lisaa_mitta
BEGIN
    alkiot_lkm = alkiot_lkm+1 -- seuraavaksi lisättävän mitan numero
    -- Jos taulukko ei ole vielä täynnä, tallennetaan uusi arvo
viimeiseksi
    IF alkiot_lkm <= pituus THEN
        taulukko[alkiot_lkm]=mitta
    ELSE
        -- Taulukko on täynnä, siirretään arvoja yhden pykälän verran
        FOR i = 2 to pituus DO
            taulukko[i-1]=taulukko[i]
        ENDFOR
        -- Tallennetaan uusin arvo taulukon viimeisimmäksi
        taulukko[pituus]=mitta
    ENDIF
END lisaa_mitta

ROUTINE lue_tiedot
BEGIN
    tiedosto = muistilaite + tiedosto
    OPEN FILE lokitiedosto('RO', tiedosto)

    IF (IO_STATUS(lokitiedosto) <> 0) or (kohde_valmis=FALSE) THEN
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe tiedoston avaamisessa!')
            ABORT
            PAUSE
        PAUSE
        ABORT
    ENDIF
```

```
    alkiot_lkm = 0
    REPEAT
        rivi = ''
        -- Lue rivi
        -- Lue ensimmäiseen pilkkuun asti
        REPEAT
            READ lokitiedosto (merkki::1::0)
            IF IO_STATUS(lokitiedosto)=2021 THEN
                GO TO loppu
            ENDIF
            IF (merkki<>' ') and (merkki<>',') THEN
                rivi = rivi + merkki
            ENDIF
        UNTIL (merkki=',') or (merkki=chr(13)) or (merkki=chr(10))
        REPEAT
            READ lokitiedosto (merkki::1::0) -- luetaan rivi loppuun
            IF IO_STATUS(lokitiedosto)=2021 THEN
                GO TO loppu
            ENDIF
        UNTIL (merkki=chr(13)) or (merkki=chr(10))
        CNV_STR_REAL(rivi, mitta)
        WRITE TPDISPLAY('rivi ', rivi, cr)
        WRITE TPDISPLAY('mitta', alkiot_lkm, ' - ', rivi, cr)
        IF (UNINIT(mitta)=false) THEN
            lisaa_mitta
        ENDIF
        WRITE TPDISPLAY(cr)
        loppu::
    UNTIL IO_STATUS(lokitiedosto)=2021
    CLOSE FILE lokitiedosto
END lue_tiedot

ROUTINE lue_param
BEGIN
    --    1. parametri on tiedostonnimi, pitää olla string-tyyppiä
    GET_TPE_PRM(1, datatyyppi, prm_int, prm_real, tiedosto, STATUS)
    IF ((STATUS <> 0) OR (datatyyppi <> 3)) THEN
        WRITE TPERROR (CHR(137) + CHR(128))
        WRITE TPERROR('Virhe parametrissa 1! Lue syntaksi ohjeesta.')
        PAUSE
        ABORT
    ENDIF

    --    2. parametri on käytettävä muistilaite
    --    1=robotin sisäinen muisti
    --    2=USB-muisti
    GET_TPE_PRM(2, datatyyppi, prm_int, prm_real, prm_str, STATUS)
    IF ((STATUS <> 0) OR (datatyyppi <> 1)) THEN
        WRITE TPERROR (CHR(137) + CHR(128))
        WRITE TPERROR('Virhe parametrissa 2! Lue syntaksi ohjeesta.')
        PAUSE
        ABORT
    ENDIF
    SELECT prm_int OF
        CASE(1):
            muistilaite = 'MC:'
        CASE(2):
            muistilaite = 'UD1:'
```

```
        ELSE:
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe parametrissa 2! Lue syntaksi
ohjeesta.')
            PAUSE
            ABORT
    ENDSELECT

    --    3. parametri on suoritettava tehtävä
    --    1 = yksittäinen arvo
    --    2 = aritmeettinen keskiarvo
    --    3 = keskihajonta
    --    4 = otoksen pienin arvo
    --    5 = otoksen suurin arvo
    --    6 = yhteenveto näytölle
    GET_TPE_PRM(3, datatyyppi, tehtava, prm_real, prm_str, STATUS)
    IF ((STATUS <> 0) OR (datatyyppi <> 1)) THEN
        WRITE TPERROR (CHR(137) + CHR(128))
        WRITE TPERROR('Virhe parametrissa 3! Lue syntaksi ohjeesta.')
        PAUSE
        ABORT
    ENDIF


    -- 4. parametri, määrä
    GET_TPE_PRM(4, datatyyppi, maara, prm_real, prm_str, STATUS)
    IF STATUS <> 0 THEN
        IF STATUS = 17042 THEN
        WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe! Ohjelma vaatii neljä parametria.')
        ENDIF
        PAUSE
        ABORT
    ENDIF
    IF datatyyppi <> 1 THEN
        WRITE TPERROR (CHR(137) + CHR(128))
        WRITE TPERROR('Parametrin 4 pitää olla kokonaisluku.')
        PAUSE
        ABORT
    ENDIF

    -- 5. parametri, tulosrekisteri
    IF tehtava<>6 THEN
        GET_TPE_PRM(5, datatyyppi, tulosrek, prm_real, prm_str,
STATUS)
        IF STATUS <> 0 THEN
            IF STATUS = 17042 THEN
            WRITE TPERROR (CHR(137) + CHR(128))
                WRITE TPERROR('Virhe! Ohjelma vaatii neljä
parametria.')
            ENDIF
            PAUSE
            ABORT
        ENDIF
        IF datatyyppi <> 1 THEN
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Parametrin 5 pitää olla kokonaisluku.')
            PAUSE
            ABORT
```

```
        ENDIF
    ENDIF
END lue_param

BEGIN

    lue_param
    lue_tiedot
    IF alkiot_lkm>pituus THEN
        alkiot_lkm=pituus
    ENDIF

    SELECT tehtava OF
    CASE(1):
        SET_REAL_REG(tulosrek, arvo, STATUS)
    CASE(2):
        SET_REAL_REG(tulosrek, keskiarvo, STATUS)
    CASE(3):
        SET_REAL_REG(tulosrek, hajonta, STATUS)
    CASE(4):
        SET_REAL_REG(tulosrek, minimi, STATUS)
    CASE(5):
        SET_REAL_REG(tulosrek, maksimi, STATUS)
    CASE(6):
        yhteenveto
    ELSE:
        WRITE TPERROR (CHR(137) + CHR(128))
        WRITE TPERROR('Virhe parametrissa 4! Lue syntaksi ohjeesta.')
        PAUSE
        ABORT
    ENDSELECT

END analysoi
```

LIITE 10.

OHJELMALISTAUS, KORJAIN

```
PROGRAM korjain

-- KORJAIN(korjaimen arvo)
-- Muut muuttujat asetetaan karel vars -valikossa robotilla

%COMMENT = 'Korjain sorville'
%ENVIRONMENT STRNG
%ENVIRONMENT FDEV

-- Muuttujat
VAR
    -- Käyttäjän robotilla konfiguroimat muuttujat
    skaalaus : integer
    lahto : integer
    liipaisu : integer
    etumerkki : integer
    bittimaara : integer

    -- Muut muuttujat
    korjaus : real
           prm_int : INTEGER
           prm_real : REAL
           prm_str : STRING[8]
           STATUS : INTEGER
           datatyyppi : INTEGER

ROUTINE maks_arvo : integer
VAR
    i : integer
    summa : integer
BEGIN
    summa = 1
    FOR i = 1 to bittimaara DO
        summa = summa*2
    ENDFOR
           return (summa-1)
END maks_arvo


BEGIN

    -- Luetaan korjaimen arvo
    GET_TPE_PRM(1, datatyyppi, prm_int, korjaus, prm_str, STATUS)
    IF STATUS <> 0 THEN
        IF STATUS = 17042 THEN
            WRITE TPERROR (CHR(137) + CHR(128))
            WRITE TPERROR('Virhe! Anna korjaimen arvo parametrina.')
        ENDIF
        ABORT
    ENDIF
    IF datatyyppi <> 2 THEN  -- Jos parametri ei ole liukuluku
        -- Jos parametri on kokonaisluku, sen arvo tallennetaan,
muuten palautetaan virhe
        IF datatyyppi = 1 THEN
            korjaus = prm_int
        ELSE
            WRITE TPERROR (CHR(137) + CHR(128))
```

```
            WRITE TPERROR('Parametrin 1 pitää olla luku.')
            ABORT
        ENDIF
    ENDIF

    -- Korjain muutetaan oikeaan skaalaan lähetystä varten
    korjaus = korjaus*skaalaus

    -- Jos korjain on negatiivinen, asetetaan etumerkkibitti
    IF korjaus < 0 THEN
        DOUT[etumerkki] = true
    ENDIF

    -- Jos korjain on ylisuuri, rajoitetaan sen koko ryhmälähdön
bittisyyden mukaan
    IF (korjaus > maks_arvo) OR (korjaus < maks_arvo*(-1)) THEN
        korjaus = maks_arvo
        WRITE TPERROR (CHR(137) + CHR(128))
        WRITE TPERROR('Korjainta ei lähetetty kokonaisena.')
    ENDIF

    -- Asetetaan ryhmälähdön tila
    -- Korjaus pyöristetään ensin kokonaisuluvuksi, negatiivisuus
otetaan pois itseisarvolla
    GOUT[lahto]=ABS(ROUND(korjaus))

    DOUT[liipaisu] = true

    DELAY(250)

    DOUT[liipaisu] = false
    DOUT[etumerkki] = false

END korjain
```

LIITE 11.

OHJELMALISTAUS MAKRO-OHJAUS SORVILLE

```
%
O6100(KORJAIN)
(1 KORJATTAVAN TYOKALUN NUMERO)
(2 VALIAIKAINEN PAIKALLINEN MUUTTUJA)

(TARKISTETAAN PARAMETRI)
IF [#1 LT 1] GOTO 999
IF [#1 GT 64] GOTO 999

(VARMISTETAAN ETTA NOLLAUSBITTI ON ALHAALLA)
#1104=0

(LASKETAAN KORJAIMEN ARVO)
#2=#1005
#2=#2+#1006*2
#2=#2+#1007*4
#2=#2+#1008*8
#2=#2+#1009*16
#2=#2+#1010*32
(MUUTETAAN ETUMERKKI TARVITTAESSA)
IF [#1011 EQ 0] GOTO 2
#2=#2*[-1]
N2

(SKAALATAAN KORJAIN -63...63)
(ALUEELLE -0,063...0,063)

#2=#2/1000

(ANNETAAN LOGIIKALLE NOLLAUSTIETO)
#1104=1

(ASETETAAN KORJAINARVO HALUTULLE TERALLE)
(TERAKORJAIMET ALKAVAT MUISTIPAIKASTA 2001)
#[2000+#1]=#[2000+#1]+#2
G4X0.1
(NOLLAUSBITTI ALAS)
#1104=0

M99

N999 (VIRHE)
#3000=1(VIRHE PARAMETRISSA)
M99%
```

LIITE 12.

KÄYTTÖOHJEET

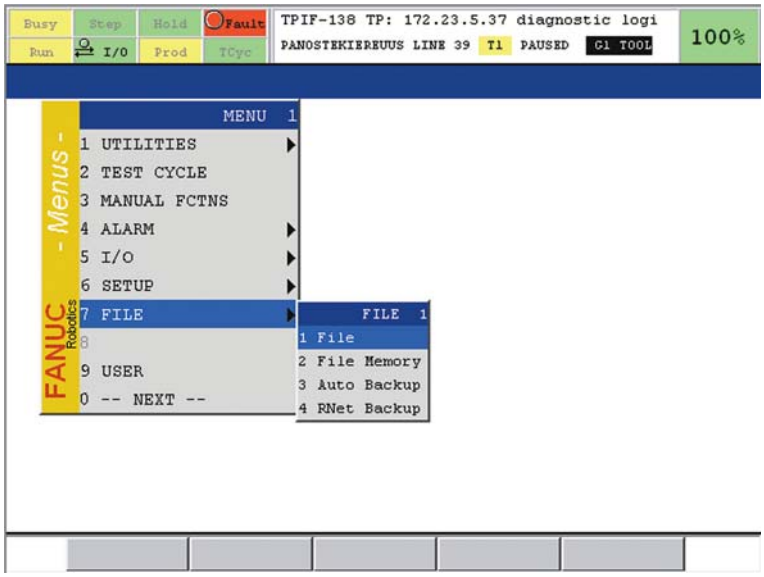
1 Ohjelmien asentaminen

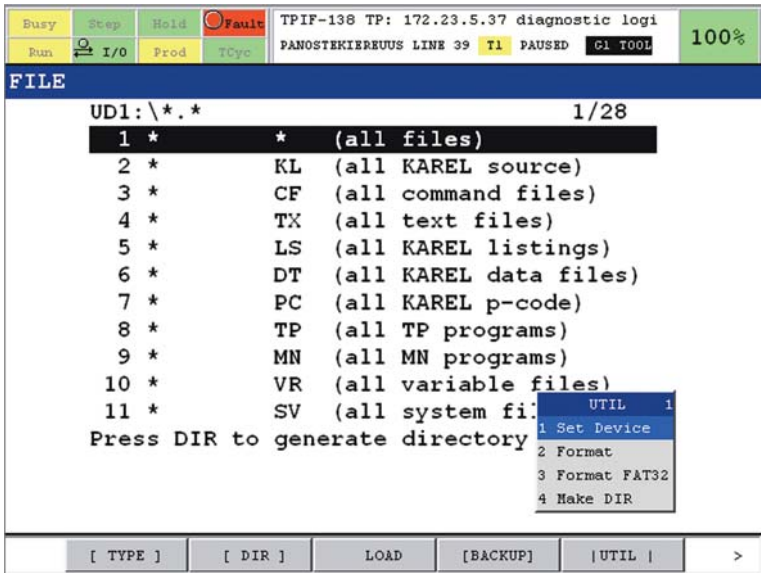
Kokonaisuuteen kuuluu kuusi robotille asennettavaa ohjelmaa: Keyence-mittalaitteen rajapinta, Mitutoyo-mittalaitteen rajapinta, ohjelmallinen hyväksy/hylkää-tulkki, tietoja tiedostoon kirjaava moduuli, tietojen analysointimoduuli ja teräkorjaimen sorville lähettävä ohjelma.

Ohjelmat on tehty Karel-ohjelmointikielellä. Ne pitää ennen käyttöä ladata robottiohjaimen muistiin. Tiedostoja voi ladata CF-muistikortilla, USB-muistitikulla tai verkon kautta. Tässä esimerkissä tiedostot siirretään tavallisen USB-muistitikun avulla.

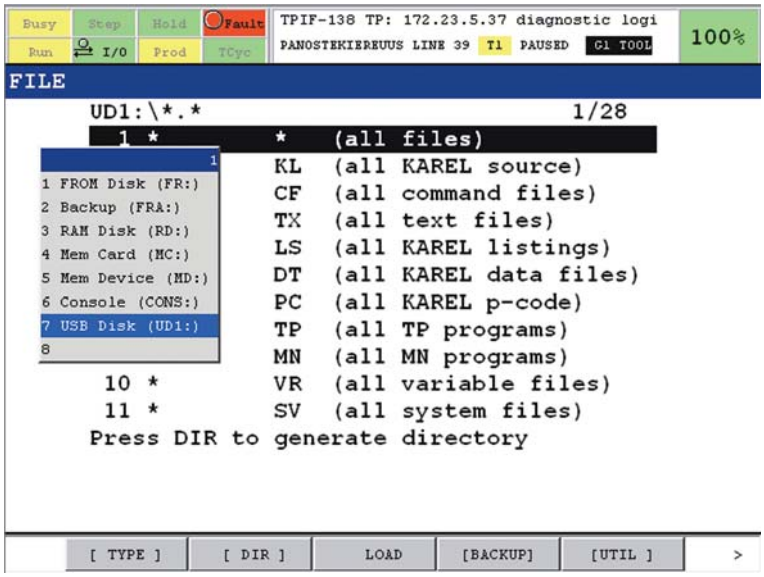
Aloita lataamalla halutut ohjelmat muistitikulle. Ohjelmien tunnisteena on .pc, esimerkiksi tulkki.pc. Tikulla saa olla muitakin tiedostoja, sitä ei tarvitse tyhjentää tiedostojen robotille lataamista varten.

Kun USB-muistitikku on liitetty robotin ohjauspaneelissa olevaan USB-liittimeen, paina MENU, valitse kuvan mukaisesti FILE-valikosta kohta FILE ja hyväksy enterillä.

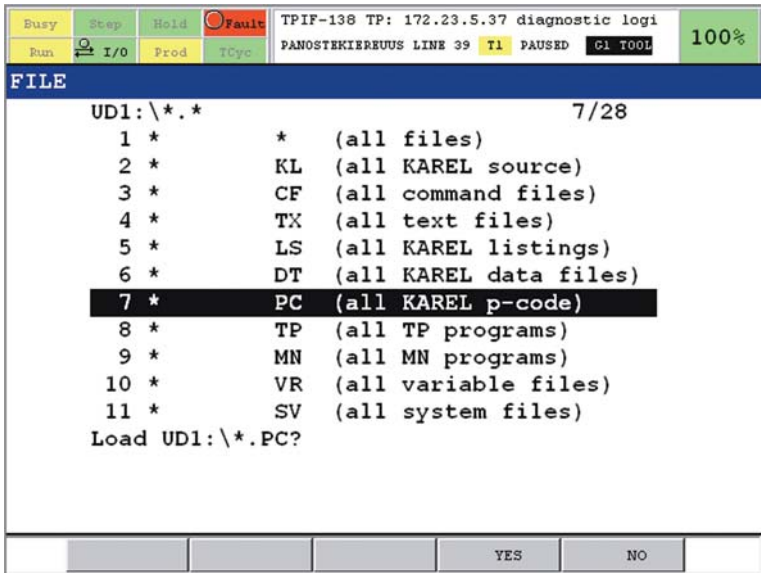




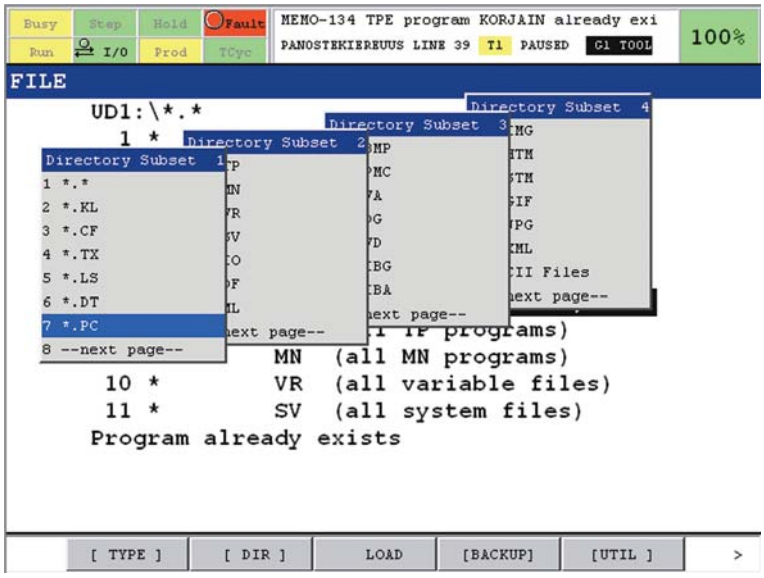
Varmista, että muistilaitteena käytetään USB-tikkua. Valitse UTIL (F5) ja listasta Set Device.



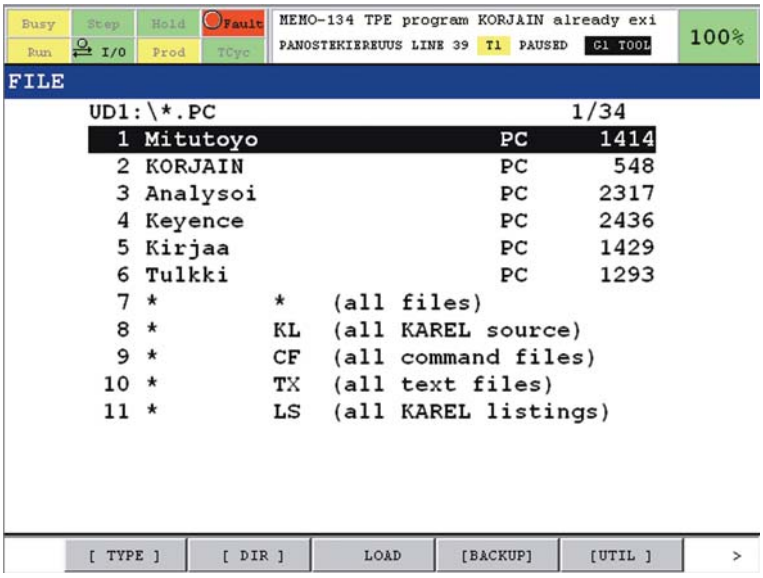
Aukenevasta listasta valitaan USB Disk (UD1:). Nyt robotilla voidaan selata USB-muistitikun tiedostoja ja ladata niitä robottiohjaimen muistiin.



Nopein tapa ladata kaikki muistitikulla olevat käännetyt Karel-tiedostot on valita listasta kohta PC ja ladata (LOAD) ne robotille näppäimellä F3. Latauskäsky pitää vielä hyväksyä vastaamalla YES näppäimellä F4. Tämä lataa USB-muistitikulta kaikki .pc-tiedostot robotin muistiin. Muistitikulla saa olla muitakin tiedostoja, se ei haittaa lataamista.



Jos on tarpeen ladata vain yksi tiedosto useasta, voi valita näppäimellä F2 kohdan DIR. Tällä valinnalla voi rajata tiedostolistauksen tiedostotyypin mukaan. Valitse listasta *.PC ja hyväksy enterillä.



Ruudulle tulee kaikki aktiivisella muistilaitteella olevat .pc-tiedostot. Valitse haluamasi tiedosto ja lataa se robotille valinnalla LOAD (F3).

Nyt kaikki tarvittavat ohjelmat on ladattu robotille ja ne ovat valmiina käytettäväksi.

Vielä pitää tarkistaa, että robotin järjestelmäparametreissa on valittu Karel-ohjelmat käyttöön. Järjestelmämuuttujassa `$karel_enb` pitää olla arvo `1`, jotta Karelilla tehdyt ohjelmat saa näkyviin.

Ole varovainen, kun tarkistat järjestelmämuuttujia. Älä koske mihinkään, minkä toiminnasta et ole varma, koska järjestelmämuuttujien väärällä konfiguroinnilla on mahdollista sotkea robotin toiminta. Järjestelmämuuttujiin pääset käsiksi painamalla MENU, valitsemalla seuraavalta sivulta SYSTEM ja sieltä VARIABLES. Muuta tarvittaessa `$karel_enb` arvoksi `1` ja poistu valikosta. Nyt ohjelmia voi käyttää.

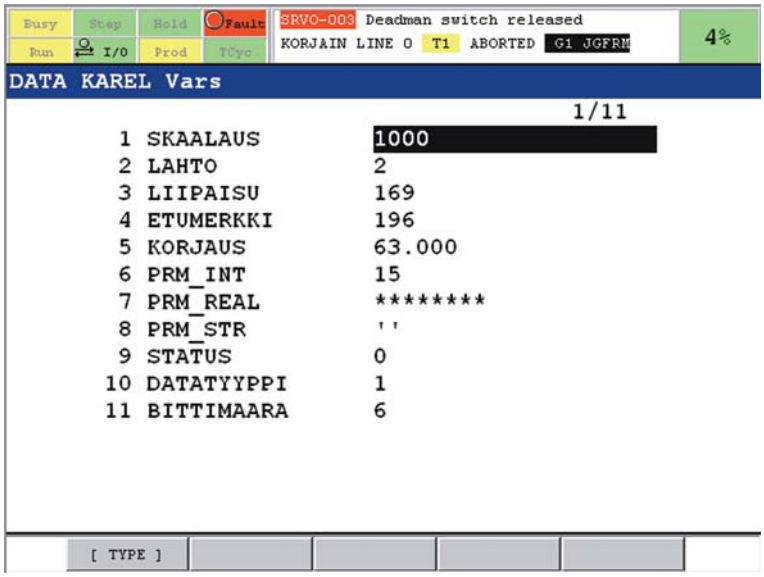
2 Asetusten muuttaminen

Kaikki muut ohjelmat toimivat sellaisenaan, mutta korjaimen sorville lähettävä ohjelma pitää konfiguroida ennen käyttöönottoa. Se on helppoa, tarvitaan ainoastaan robotin ryhmälähdön asettaminen ja lisäksi Karel-ohjelmalle pitää kertoa viisi asiaa:

- mikä lähtö toimii liipaisimena tiedonsiirrolle
- mikä lähtö välittää etumerkkitiedon
- mitä ryhmälähtöä käytetään tiedonsiirtoon
- miten monta bittiä tiedonsiirtoon on varattu
- mitä skaalauskerrointa käytetään (oltava sama kuin sorvin makrossa).

Robotin ryhmälähtö konfiguroidaan GROUP I/O -valikosta. Tukeudu tässä tarvittaessa robottiohjauksen käyttöoppaaseen.

Korjaimen lähettävän ohjelman asetukset muutetaan Karel-muuttujavalikosta. Ennen kuin muuttujia voi käsitellä, pitää ohjelma aktivoida. Mene ohjelmavalikkoon SELECT-näppäimellä ja selaa kursori korjain-ohjelman päälle. Paina ENTER ja ruudun alariville tulee tieto "Korjain is selected". Nyt ohjelma on valittu. Mene DATA-napilla rekisterivalikkoon ja valitse type (F1) -valikosta kohta "Karel Vars". Tässä valikossa on aktiivisen ohjelman muuttujat.



Tässä esimerkissä skaalauksen arvo on 1000, ryhmälähdön numero on 2, liipaisubitti on DO169, etumerkkibitti on DO196 ja ryhmälähdölle varattu bittimäärä on 6.

3 Käyttäminen

Kaikkia ohjelmia koskevat samat yleisperiaatteet. Niitä käytetään robotin työohjelmassa aliohjelmakutsuilla. Ohjelmille annetaan kutsun yhteydessä lisätietoja parametreina. Parametrit voidaan syöttää suoraan tai lukea rekistereistä joko suorilla tai epäsuorilla viittauksilla.

Tapauksesta riippuen käytetään kahdenlaisia aliohjelmakutsuja: CALL ja RUN. Näiden välinen ero on siinä, että CALL-kutsulla käynnistetty ohjelma suoritetaan kokonaisuudessaan loppuun ennen robotin työohjelman jatkamista, kun taas RUN-käskyllä kutsuttu ohjelma suoritetaan rinnakkaisajona robotin työohjelman kanssa. RUN-kutsulla ohjelma siis pyörii taustalla ja robotti voi jatkaa omaa toimintaansa välittömästi ohjelmakutsun jälkeen, vaikka KAREL-ohjelma pyörisikin taustalla.

Robotin ohjausjärjestelmässä on eräs rajoitus koskien aliohjelmakutsuja. CALL-kutsuun on mahdollista antaa parametreja ohjaamaan aliohjelman toimintaa, mutta RUN-kutsussa tämä ei onnistu. Tämä rajoitus kierretään tässä tapauksessa kapseloimalla aliohjelma erilliseen työohjelmaan, jossa ainoana asiana käynnistetään oikea Karel-ohjelma CALL-komennolla sopivilla parametreilla. Tämä kapseloitu aliohjelma kutsutaan varsinaisessa työohjelmassa RUN-komennolla, jolloin ohjelma saadaan tausta-ajoon, mutta sille on kuitenkin määritelty ohjausparametrit.

3.1 Keyence

Ohjelmakutsun syntaksi:

```
CALL KEYENCE(portti, tehtävä, parametri, [kättelyrekisteri])
```

Portti (sallitut arvot 1 ja 2) määrittelee, kumpaan sarjaliikenneporttiin mittalaite on kytketty:

1. portti ohjauskaapin etupaneelissa
2. portti ohjauskaapin sisällä

Tehtävä (sallitut arvot 1, 2, 3 ja 4) määrittelee suoritettavan tehtävän:

1. lue mitta
2. lue alamitta vaaputtamalla kappaletta
3. muuta keskiarvoistusta
4. vaihda ohjelma

Tapauskohtainen parametri määrittelee eri toiminnoilla eri asioita,

- mittaus: rekisteri, johon tulos palautetaan (sallitut arvot: positiivinen kokonaisluku)
- suodatuksen arvo (sallitut arvot: kokonaisluku välillä 0–12)
  parametri-suodatusarvoparit:

0→1	7→128
1→2	8→256
2→4	9→512
3→8	10→1024
4→16	11→2048
5→32	12→4096Ð
6→64	

- ohjelman numero (sallitut arvot: kokonaisluku välillä 0–15)

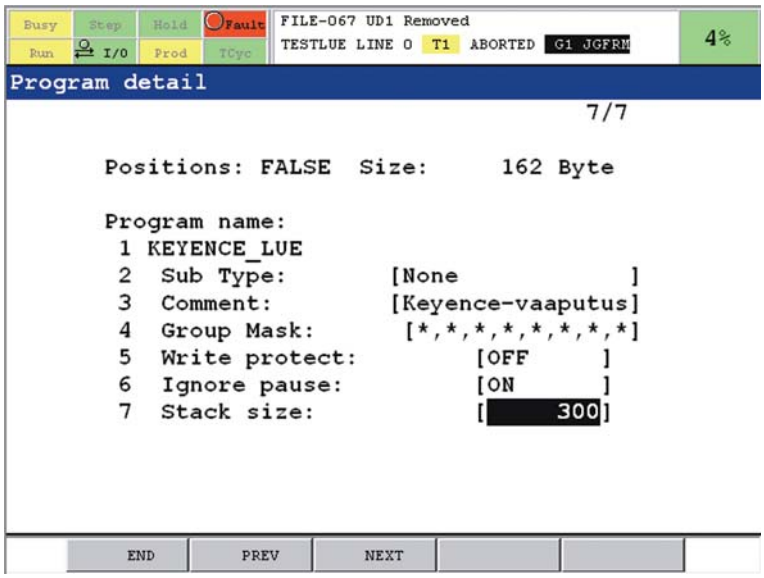
Kättelyrekisteri on ehdollinen parametri, jota käytetään vain, kun mitataan alamittaa kappaletta vaaputtamalla:

- kertoo, mitä rekisteriä kättelyyn käytetään (sallitut arvot: positiivinen kokonaisluku)

kättelyrekisterin tila ohjelman eri vaiheissa:

1: Karel → **työohjelma**: robotti saa aloittaa mittausliikkeen
2: työohjelma → **Karel**: mittausliike tehty
3: Karel → **työohjelma**: mittaustulos palautettu rekisteriin, saa poistua

Kaikki muut toiminnot onnistuvat suoraviivaisesti CALL-kutsulla, mutta alamitan lukeminen kappaletta vaaputtamalla vaatii Karel-ohjelman tausta-ajoa, koska siinä robotti liikuttaa kappaletta mittauksen aikana. Tässä pitää kapseloida ohjelmakutsu erilliseen aliohjelmaan, joka suoritetaan työohjelmassa RUN-kutsulla.



Tee ohjelma esimerkiksi nimellä KEYENCE_LUE. Tälle ohjelmalle ei tarvitse määrittää liikeryhmiä. Lisäksi kannattaa yllä olevan kuvan mukaisesti laittaa *Ingore pause* päälle. Tämä tarkoittaa käytännössä sitä, että ohjelman suoritus jatkuu taustalla, vaikka robotti seisautetaan tai käsiajolla irroitetaan sallintakytkimestä/shift-napista.

Tähän ohjelmaan tehdään CALL-kutsu varsinaiseen Keyence-rajapintaan halutuilla parametreilla. Ohjelman sisältö voi olla esimerkiksi

```
CALL KEYENCE(1,1,2,3)
```

Nyt voit varsinaisessa työohjelmassa kutsua RUN KEYENCE_LUE, jolloin tausta-ajona käynnistyy alamitan lukeminen etupaneelin sarjaportista niin, että rekisteriin 2 palautetaan mittatulos ja rekisteriä 3 käytetään kättelyyn.

Esimerkki vaaputtamalla tehtävästä mittauksesta:

```
L P[2] 100mm/sec FINE
  RUN KEYENCE_LUE
  WAIT R[3: Kattely ]=1
L P[3] 5 deg/sec FINE
  R[3: Kattely ]=2
  WAIT R[3: Kattely ]=3
```

Tässä esimerkissä pyöritään robotin työkalupisteen ympäri. Käytännössä on järkevää tehdä mittalaitetta varten ulkoinen työkalupiste, jonka suhteen vaaputtamalla mittausliike tehdään. Näin erilaisten kappaleiden mittausliikkeen ohjelmointi on hyvin suoraviivaista. Katso tarvittaessa robotin käyttöohjeesta neuvoja ulkoisen työkalupisteen määrittelemiseen ja käyttämiseen.

3.2 Mitutoyo

Ohjelmakutsun syntaksi:

```
CALL MITUTOYO(portti, tulosrekisteri, kättelyrekisteri)
```

Portti (sallitut arvot 1 ja 2) määrittelee, kumpaan sarjaliikenneporttiin mittalaite on kytketty:

1. portti ohjauskaapin etupaneelissa
2. portti ohjauskaapin sisällä

Mittausrekisteri on rekisteri, johon tulos palautetaan (sallitut arvot: positiivinen kokonaisluku)

Kättelyrekisteri kertoo, mitä rekisteriä kättelyyn käytetään (sallitut arvot: positiivinen kokonaisluku)

kättelyrekisterin tila ohjelman eri vaiheissa:

1: Karel → **työohjelma**: robotti saa aloittaa mittausliikkeen
2: työohjelma → **Karel**: mittausliike tehty
3: Karel → **työohjelma**: mittaustulos palautettu rekisteriin, saa poistua

Koska Mitutoyolla mittaus tehdään pyyhkäisevällä liikkeellä, pitää ohjelma suorittaa tausta-ajona RUN-käskyllä. Keyence-kohdassa on kerrottu, miten ohjelmakutsu kapseloidaan aliohjelmaan. Mitutoyon kohdalla toimitaan samoin.

Esimerkki sisämittauslaitteella tehtävästä mittauksesta:

```
L P[2] 100mm/sec FINE (viedään kappale mittalaitteelle)
  RUN MITUTOYO_LUE
  WAIT R[3: Kattely ]=1
L P[3] 20 mm/sec FINE (pyyhkäisevä liike, pisteitä voi olla
R [3: Kattely ]=2 tarvittaessa useitakin)
  WAIT R[3: Kattely ]=3
```

3.3 Tulkki

Ohjelmakutsun syntaksi:

```
CALL TULKKI(mitta, alaraja, yläraja, tulosrekisteri,
[epävarmuus])
```

Tulkin avulla voidaan tarkastaa, onko mitta sallituissa rajoissa. Lisäksi tulkki ottaa mittalaitteen epävarmuuden tarvittaessa huomioon.

Ohjelmakutsussa mitta ja rajat ovat kokonais- tai liukulukuja, jotka voidaan syöttää suoraan numeroarvoina tai rekisteriviittauksina. Käytännössä on todennäköisesti helpointa syöttää ala- ja ylärajat suorina numeroarvoina, kun taas mitta-arvo tulee otetaan rekisteristä, minne mittalaiterajapinta on sen tallentanut.

Epävarmuus on vapaavalintainen parametri, sen voi halutessaan jättää pois.

Tulosrekisteriin palautettavat arvot:

1	Mitta on toleranssialueella.
-1	Mitta on toleranssialueen yläpuolella.
-2	Mitta on toleranssialueen alapuolella.

Esimerkki tulkin käytöstä

```
CALL TULKKI(R[4: Mitta ], 15.97, 16.03, 6, 0.002)
```

Mittatulos haetaan rekisteristä 4. Tuloksen pitää olla välillä 15,97 - 16,03. Epävarmuus mittalaitteessa on 0,002 mm, joka vähennetään toleranssialueesta. Tulkin tulos palautetaan rekisteriin 6. Jos tulkin jälkeen rekisterin arvo on negatiivinen, voidaan esimerkiksi pysäyttää robotin ohjelma ja hälyttää operaattori paikalle tai pudottaa kappale hylkylaatikkoon ja jatkaa tuotantoa normaalisti.

3.4 Kirjaaminen

Tietojen kirjaaminen on tämän kokonaisuuden yleiskäyttöisin komponentti. Sitä voi käyttää moneen muuhunkin asiaan kuin mittauspöytäkirjan tekemiseen.

Kirjaamisohjelman syntaksi:

```
CALL KIRJAA(tiedosto, muistilaite, mitta, lisätieto 1 …
lisätieto 7)
```

Tiedosto määrittelee, minkä nimiseen tiedostoon kirjoitetaan. Tiedostopäätteen voi määritellä itse, ohjelma ei aseta sen suhteen rajoituksia. Koska tiedostot ovat sisällöltään CSV-muotoisia, on ehkä mielekästä käyttää myös tiedostopäätteenä .csv:tä.

Muistilaite (sallitut arvot 1 ja 2) määrittelee, minne tiedosto tallennetaan:

1. CF-muistikortti ohjauskaapin sisällä
2. USB-muistilaite ohjauskaapin paneelissa

Kun kirjataan mittatietoja analysointia varten, on mittatiedon aina oltava ensimmäisenä kirjattavista tiedoista. Analysointimoduuli olettaa, että mittatieto on ensimmäisessä kentässä.

Loppuihin parametreihin voi kirjata mitä tahansa. Kirjattava tieto voidaan antaa suoraan ohjelmakutsun yhteydessä kiinteänä tietona tai se voi tulla rekistereistä. Esimerkiksi tämän kappaleen jälkeen annettu korjainarvo saattaa olla mielenkiintoinen lisätieto.

Päivämäärä ja kellonaika kirjoitetaan jokaisen rivin perään automaattisesti.

3.5 Analysoi

Analysointimoduulilla voidaan lukea tiedostosta mittoja ja ottaa niistä yksinkertaisia ja yleiskäyttöisiä tilastollisia tunnuslukuja. Lisäksi ohjelmalla voi tuottaa yhteenvetonäkymän mittauksista robotin näytölle.

Analysointiohjelman syntaksi:

CALL ANALYSOI(tiedosto, muistilaite, tehtävä, määrä, tulosrekisteri)

Tiedosto määrittelee, minkä nimisestä tiedostosta analysoitava tieto haetaan.

Muistilaite (sallitut arvot 1 ja 2) määrittelee, mistä tiedosto haetaan:

1. CF-muistikortti ohjauskaapin sisällä
2. USB-muistilaite ohjauskaapin paneelissa

Tehtävä (sallitut arvot 1, 2, 3, 4, 5 ja 6) määrittelee suoritettavan tehtävän:

1. yksittäinen arvo
2. aritmeettinen keskiarvo
3. keskihajonta
4. otoksen pienin arvo
5. otoksen suurin arvo
6. yhteenveto USER-näkymään

Määrä (sallittu arvo: positiivinen kokonaisluku) määrittelee, miten monen kappaleen otos otetaan (tai miten mones mitta palautetaan, jos tehtävänä on palauttaa yksittäinen mitta).

Tulosrekisteri määrittelee, mihin rekisteriin tulos palautetaan. Tätä parametria ei tarvita, jos tehtävänä on 6 eli yhteenveto näytölle.

Esimerkki analysoinnin käytöstä

```
CALL ANALYSOI('esim.csv', 2, 2, 10, 7)
```

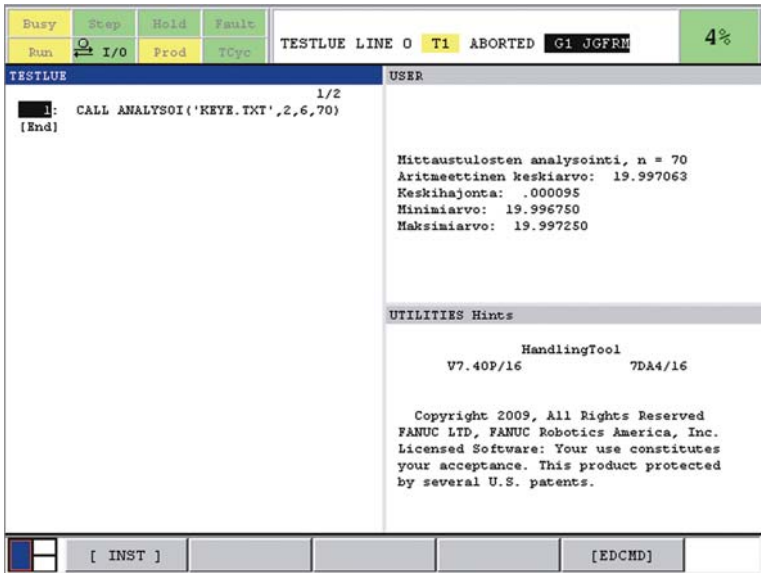
Haetaan USB-muistitikulta tiedostosta esim.csv kymmenen edellisen mittaustuloksen aritmeettinen keskiarvo rekisteriin 7.

```
CALL ANALYSOI('esim.csv', 2, 4, 3, 8)
```

Haetaan USB-muistitikulta tiedostosta esim.csv kolmen edellisen mittaustuloksen minimi rekisteriin 8.

```
CALL ANALYSOI('KEYE.TXT', 2, 6, 70)
```

Haetaan USB-muistitikulta tiedostosta KEYE.TXT 70 edellisen mittaustuloksen yhteenveto näytölle. Kuvassa näkyy esimerkki yhteenvedosta. Yhteenveto näytetään aina USER-näkymässä, jonka käyttäjä voi avata haluamaansa ikkunaan tai koko näyttöön valikosta MENU kohdasta USER.



3.6 Korjain

Korjainohjelman syntaksi:

```
CALL KORJAIN(korjain)
```

Korjain annetaan etumerkkeineen millimetreinä. Jos korjain on suurempi kuin käytössä oleva ryhmälähtö kykenee siirtämään, siirretään suurin mahdollinen korjain ja ilmoitetaan käyttäjälle näytön yläreunan tilarivillä tapahtuneesta. Ohjelmaa ei keskeytetä.

4 Esimerkki kokonaisuuden käyttämisestä

Toimintaketju kappaleen mittaamisesta teräkorjaimen lähettämiseen toimii eri moduulien yhteiskäytöllä. Yleistasolla tämä tapahtuu esimerkiksi seuraavalla tavalla (rekisterit ovat esimerkinomaisia, käytettävillä osoitteilla ei ole merkitystä):

Luetaan kappaleen mitta muistirekisteriin R100.

Lasketaan teräkorjaimen arvo valmiiksi rekisteriin R101. Korjaimen arvona käytetään neljän edellisen kappaleen keskiarvon ja nimellismitan erotusta. (Analysoi, keskiarvo, rekisteriin R102)

Laskettu korjain lähetetään sorville kuitenkin vain siinä tapauksessa, että neljä peräkkäistä mittausta on ollut alle tai yli nimellismitan. (Analysoi, minimi, rekisteriin R103; analysoi, maksimi, rekisteriin R104). Jos R103>nimellismitta tai R104<nimellismitta, kutsutaan korjain ja annetaan parametriksi R101. Jos korjainta ei anneta, nollataan R101 siksi, että tiedostoon kirjataan tehdyksi korjaimeksi 0, jos korjainta ei annettu.

Tämän jälkeen kappaleen mitta vielä tarkistetaan. (Tulkki, rekisteriin R105)

Kappale viedään oikeaan paikkaan tulkin tuloksen perusteella.

Kirjataan lokitiedostoon kappaleen mittaustulos R100. Kappaleen jälkeen annettu korjainarvo on R101 ja tulkin tulos R105.